Il Mining e Il Consenso

Introduzione

Il Mining è il processo in cui nuovi bitcoin sono aggiunti al totale di moneta in circolazione. Il Mining inoltre serve per proteggere il sistema bitcoin contro transazioni fraudolente o transazioni che cercano di spendere lo stesso numero di bitcoin più di una volta, problema conosciuto con il nome di double-spend (doppia-spesa). I miner provvedono alla potenza di calcolo per il network bitcoin in cambio dell'opportunità di essere ricompensati con dei bitcoin.

I miner validano nuove transazioni e le registrano sulla ledger globale (un libro mastro globale). Un nuovo blocco, contenente le transazioni che sono avvenute dopo la scoperta dell'ultimo blocco, è minato in media ogni 10 minuti, e inoltre accodando queste transazioni alla blockchain. Le transazioni che sono diventate parte di un blocco e aggiunte alla blockchain sono considerate "confermate", questo permette ai nuovi proprietari dei bitcoin di spendere i bitcoin ricevuti nelle suddette transazioni.

I miner ricevono due tipi di ricompense per il lavoro di mining: i nuovi bitcoin creati ogni nuovo blocco, e le commissioni (fee) di transazione da tutte le transazioni incluse nel blocco. Per ottenere questa ricompensa, i miner devono competere nel risolvere un difficile problema matematico basato su un algoritmo crittografico di hashing. La soluzione al problema, chiamato proof of work, è incluso nel nuovo blocco e agisce come prova che il miner ha impiegato uno sforzo computazionale adeguato. La gara a risolvere l'algoritmo di proof-of-work per ottenere la ricompensa e il diritto di registrare le transazioni nella blockchain è alla base del modello di sicurezza di bitcoin.

Il processo di generazione di nuova moneta è chiamato mining perchè la ricompensa è fatta per simulare il rendimento decrescente, con lo stesso funzionamento dell'estrazione (mining - ndt) di metalli preziosi. La massa monetaria di Bitcoin è creata attraverso appunto questa estrazione, il mining, nello stesso modo in cui una banca centrale emette nuova moneta con la stampa delle banconote. La somma di nuovi bitcoin creati che un miner può aggiungere in un blocco decresce approssimativamente ogni quattro anni (o precisamente ogni 210.000 blocchi). E' iniziata con 50 bitcoin per blocco nel Gennaio del 2009 e si è dimezzata a 25 bitcoin per blocco nel Novembre del 2012. Si dimezzerà nuovamente a 12.5 bitcoin per blocco a qualche punto nel 2016 (probabilmente verso ottobre n.d.t.). Basato su questa formula, la ricompensa del mining di bitcoin decresce esponenzialmente fino approssimativamente all'anno 2140, quando tutti i bicoin (20,99999998 millioni) saranno stati emessi. Dopo il 2140, nessun nuovo bicoin verrà emesso.

I miner bitcoin inoltre guadagnano un compenso dalle transazioni. Ogni transazione può includere una fee di transazione, nella forma di un surplus di bitcoin tra gli input e gli output della transazione. Il miner bitcoin vincente può "tenere il resto" della transazione inclusa nel blocco vincente. Ad oggi, le fee rappresentano 0.5% o meno delle entrate di un miner bitcoin, la maggior parte derivanti dal conio dei nuovi bitcoin. In ogni caso, visto che la ricompensa diminuisce nel tempo e il numero di transazioni per blocco aumenta, una maggiore proporzione degli introiti ottenuti con il mining di bitcoin proverrà dalle fee. Dopo il 2140, tutti i guadagni dei miner bitcoin saranno nella forma di fee di transazione.

La parola "mining" è in qualche modo ingannevole. Evocando l'estrazione di metalli preziosi,

concentra la nostra attenzione nella ricompensa del mining, i nuovi bitcoin in ogni nuovo blocco. Anche se il mining è incentivato da questa ricompensa, il primo scopo del mining non è la ricompensa o la generazione di nuova moneta. Se vedi il mining solo come il processo nel quale la moneta è creata, ne stai confondendo i mezzi (gli incentivi) come se fossero il traguardo del processo. Il mining è il processo centrale della clearing house (camera di compensazione bancaria), col quale le transazioni sono validate ed evase. Il mining rende protetto (da attacchi informatici ndt.) il sistema bitcoin e abilita l'emergenza di un consenso su tutta la rete senza il bisogno di un'autorità centrale.

Il mining è l'invenzione che rende bitcoin speciale, un meccanismo di sicurezza che è la base del contante digitale peer-to-peer. La rincompensa per le nuove monete coniate e le fee di transazione sono uno schema di incentivi che allinea le azioni dei miner con la sicurezza della rete, e simultaneamente implementa la fornitura monetaria.

In questo capitolo, per iniziare esamineremo il mining come sistema di fornitura monetaria e poi osserveremo la funzione più importante del mining: il meccanismo di consenso decentralizzato emergente che è alla base della sicurezza di bitcoin.

L'economia di Bitcoin e la Creazione di Valuta

I bitcoin sono "coniati" durante la creazione di ogni blocco con un andamento prefissato e in diminuzione. Ogni blocco, generato in media ogni 10 minuti, contiene bitcoin completamente nuovi, creati dal niente. Ogni 210.000 blocchi o approssimativamente ogni quattro anni, il tasso di emissione di moneta è abbassato del 50%. Per i primi quattro anni di vita del network, ogni blocco conteneva 50 nuovi bitcoin.

Nel Novembre 2012, il nuovo di tasso di emissione di bitcoin è stato diminuito a 25 bitcoin per blocco e diminuirà ancora a 12.5 bitcoin al blocco 420.000, che sarà scoperto in qualche punto del 2016 (circa a settembre ndt.). Il tasso di creazione di nuove monete scende esponenzialmente in questo modo in 64 "dimezzamenti" fino al blocco 13,230,000 (minati approssimativamente nell'anno 2137), quando arriverà ad avere il valore minimo di unità di 1 satoshi. Finalmente, dopo 13.44 milioni di blocchi, approssimativamente nel 2140, circa 2.099.999.997.690.000 satoshi, o circa 21 millioni di bitcoin, saranno stati emessi. In seguito, i blocchi non conterranno più nuovi bitcoin, e i miner saranno ricompensati solamente attraverso le commissioni (fee) di transazione. La riserva monetaria di bitcoin nel tempo basata su di un ritmo di emissione geometricamente decrescente mostra il totale dei bitcoin in circolazione del tempo, e il decrescente andamento d'emissione della moneta.

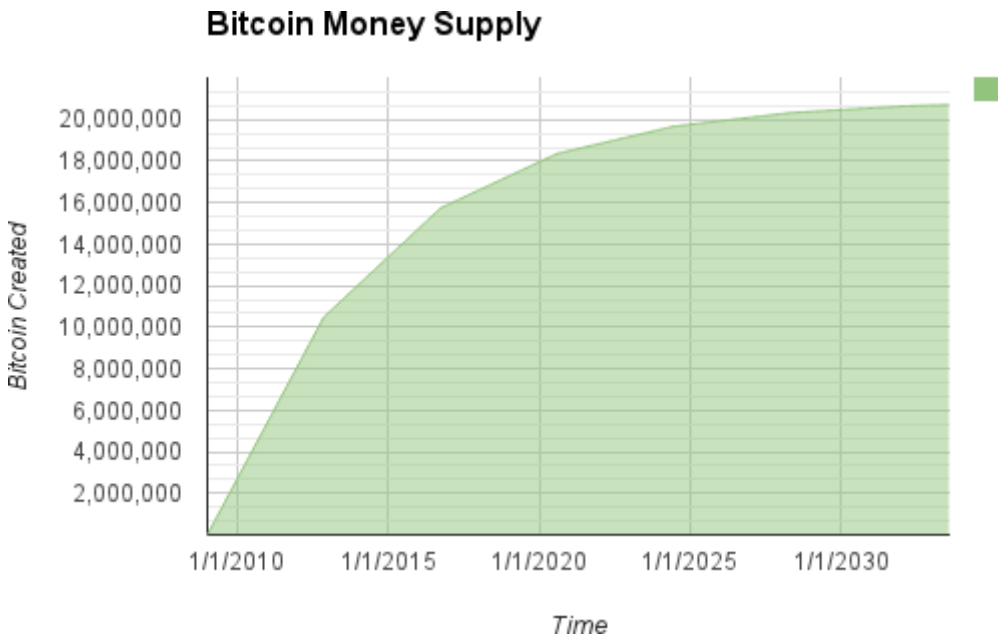

Figure 1. La riserva monetaria di bitcoin nel tempo basata su di un ritmo di emissione geometricamente decrescente

NOTE Il numero massimo di monete estratto è il *limite massimo* dei possibili premi minabili per bitcoin. In pratica, un minatore può intenzionalmente estrarre un blocco prendendo meno della piena ricompensa (che gli spetterebbe). Tali blocchi sono già stati estratti e altri potrebbero essere estratti in futuro, determinando una minore emissione totale della valuta.

Nel codice di esempio in Uno script per calcolare quanti bitcoin totali saranno emessi, calcoliamo il numero totale di bitcoin che saranno mai emessi.

Example 1. Uno script per calcolare quanti bitcoin totali saranno emessi

```
# Original block reward for miners was 50 BTC
start_block_reward = 50
# 210000 is around every 4 years with a 10 minute block interval
reward_interval = 210000

def max_money():
    # 50 BTC = 50 0000 0000 Satoshis
    current_reward = 50 * 10**8
    total = 0
    while current_reward > 0:
        total += reward_interval * current_reward
        current_reward /= 2
    return total

print "Total BTC to ever be created:", max_money(), "Satoshis"
```

Eseguendo lo script max_money.py mostra l'output prodotto eseguendo questo script.

Example 2. Eseguendo lo script max_money.py

```
$ python max_money.py
I BTC totali che saranno mai creati:  2099999997690000 Satoshis
```

L'emissione limitata e decrescente crea un approvvigionamento monetario fisso che resiste all'inflazione. A differenza di una moneta fiat, che può essere stampata in numeri infiniti da una banca centrale, il bitcoin non può mai essere gonfiato con la stampa (ndt di nuovi bitcoin).

Moneta Deflazionaria

La più importante e dibattuta consequenza di una emissione monetaria fissa e in diminuzione è il fatto che la valuta tenderà ad essere intrinsecamente *deflazionaria*. La deflazione è il fenomeno di apprezzamento di del valore dato da un disaccoppiamento tra la domanda e l'offerta che aumenta il valore (e il valore di scambio) di una moneta. L'opposto dell'inflazione, la deflazione dei prezzi sta a significare che la valuta ha più potenza d'acquisto nel tempo.

Molti economisti sostengono che un'economia deflazionaria è un disastro che dovrebbe essere evitato ad ogni costo. Questo perchè in un periodo di deflazione rapida, le persone tenono a mettere da parte la moneta invece di spenderla, sperando che i prezzi cadranno. Questo fenomeno si è verificato durante il "Decennio Perduto" del Giappone, quando un totale collasso della domanda ha spinto la moneta in una spirale deflativa.

Gli esperti di bitcoin sostengono che la deflazione non è in se per se un effetto negativo. Di solito, la deflazione è associata con un collasso di domanda perché quello è l'unico esempio di deflazione che usualmente viene studiata. In una valuta fiat con la possibilità di stampare moneta ad libitum, è molto difficile entrare in una spirale deflazionaria a meno che non ci sia un completo collasso nella domanda e la mancanza di volontà di stampare nuova moneta. La deflazione in bitcoin non è causata da un collasso nella domanda ma da una massa monetaria sempre prevedibile.

In pratica, è divenuto evidente che l'istinto di mettere da parte moneta causato da una valuta deflazionaria potrebbe essere superato tramite sconti dai venditori, fino a che lo sconto superi l'istinto di accumulare moneta del compratore. Visto che il venditore è anche lui motivato ad accumulare, lo sconto diviene il prezzo d'equilibrio al quale i due istinti di accumulo combaciano. Con sconti del 30% sul prezzo in bitcoin, molti venditori bitcoin non hanno difficoltà contro l'istinto di accumulare e generare guadagni. Rimane comunque da vedere il fatto che l'aspetto deflazionario della moneta è veramente un problema quando non è spinto da una rapida retrazione economica.

Consenso Decentralizzato

Nel capitolo precedente abbiamo esaminato la blockchain, il libro mastro globale (elenco) di tutte le transazioni, che tutti nella rete bitcoin accettano come registrazione autorevole di proprietà.

Ma come possono tutti i partecipanti del network concordare su una singola unica "verità" riguardo a chi possiede cosa, senza avere il bisogno di fidarsi di nessuno? Tutti i sistemi di pagamento tradizionale dipendono su di un modello di fiducia che ha un'autorità centrale che fornisce il servizio di camera di compensazione (clearing house), praticamente verificando e liquidando tutte le transazioni. Bitcoin non ha un'autorità centrale, ma in qualche modo ogni full node ha la copia completa di un libro mastro a cui possa affidarsi usandolo come registro autoritativo. La blockchain non è stata creata da un'autorità centrale, tuttavia è assemblata indipendentemente da ogni nodo nella rete. In qualche maniera, ogni nodo del network, agendo sulle informazioni trasmesse attraverso connessioni di rete non sicure, può arrivare alla stessa conclusione e ad assemblare una copia dello stesso libro mastro come tutti gli altri nodi. Questo capitolo esamina il processo con cui la rete bitcoin ottiene un consenso globale senza aver bisogno di un'autorità centrale.

L'invenzione più importante di Satoshi Nakamoto è il meccanismo decentralizzato di *consenso emergente*. Emergente, perché il consenso non è raggiunto esplicitamente- non c'è un'elezione o un momento fisso nel quale occorre. Invece, il consenso viene creato dalla interazione asincrona di migliaia di nodi indipendenti, che seguono tutti semplici regole. Tutte le proprietà dei bitcoin, includendo la valuta, transazioni, pagamenti e il modello di sicurezza che non dipende da un'autorità centrale o di fiducia, deriva da questa invenzione.

Il consenso decentralizzato di bitcoin emerge dalla coordinazione di quattro processi che avvengono indipendentemente sui nodi del network:

- La verifica indipendente di ogni transazione, da ogni full node, basata su una lista comprensiva di criteri Aggregazione indipendente di tali transazioni in nuovi blocchi da parte dei nodi di mining, associato a calcolo dimostrato attraverso un algoritmo di proof-of-work
- Verifica indipendente dei nuovi blocchi da ogni nodo e l'assemblaggio in una catena di blocchi
- Selezione indipendente, da ogni nodo, della catena con più calcolo cumulativo, dimostrato attraverso la prova di lavoro

Nelle prossime poche sezioni, esamineremo questi processi e come essi interagiscano per creare la proprieta emergente del consenso di rete che consente a ciascun nodo bitcoin di assemblare la propria copia dell'autorevole, fidato, pubblico, registro globale.

Verifica Indipendente delle Transazioni

In [transactions], abbiamo visto come il software del portafoglio crea transazioni raccogliendo UTXO, fornendo lo script di sblocco appropriato e quindi la costruzione di nuovi output assegnati a un nuovo proprietario. La transazione risultante viene quindi inviata ai nodi adiacenti nella rete bitcoin in modo che possa essere propagata attraverso l'intera rete bitcoin.

Invece, prima di inoltrare le transazioni ai suoi vicini, ogni nodo bitcoin che riceve una transazione per prima cosa verificherà la transazione. Questo assicura che solo le transazioni valide siano propagate sulla rete, mentre le transazioni non valide saranno scartate al primo nodo che le riceve.

Ogni nodo verifica ogni transazione seguendo una lunga lista di criteri:

- La sintassi della transazione e la struttura dati devono essere corrette.
- Ne la lista degli input ne quella degli output devono essere vuote.

- La dimensione della transazione in byte è inferiore di MAX_BLOCK_SIZE.
- I valori di ogni output, come per il totale, devono essere entro il range di valori consentito (meno di 21 milioni di bitcoin, maggiori di zero)
- Nessuno degli input hanno hash=0, N=–1 (le transazioni coinbase non devono essere trasmesse).
- nLockTime è inferiore o uguale a INT_MAX.
- La dimensione della transazione in byte è maggiore o uguale a 100.
- Il numero di operazioni di firme contenute nella transazione è inferiore del limite di operazioni di firme.
- L'unlocking script (scriptSig) puo solo aggiungere numeri allo stack, e il locking script (scriptPubkey) deve corrispondere al form isStandard (questo scarta le transazioni "nonstandard").
- Deve esistere una transazione simile nella transaction pool, o in un blocco nel ramo principale.
- Per ogni input, se l'output referenziato esiste in qualsiasi altra transazione nella pool, la transazione deve essere respinta.
- Per ogni input, cercare il ramo principale e il pool di transazioni per trovare la transazione di output referenziata. Se la transazione di output manca per qualsiasi input, questa sarà una transazione orfana. Aggiungerla al pool di transazioni orfane, se una transazione corrispondente non è già nel pool.
- Per ogni input, se l'output di transazione referenziato è un output coinbase, dovrà avere almeno COINBASE_MATURITY (100) conferme.
- Per ogni input, l'output referenziato deve esistere e non può essere già stato speso.
- Utilizzando le transazioni di output referenziate per ottenere i valori di input, verificare che ciascun valore di input, nonché la somma, siano compresi nell'intervallo di valori consentito (meno di 21 milioni di monete, più di 0).
- Respingi se la somma dei valori di input è meno della somma dei valori di output.
- Rifiutare se il costo della transazione fosse troppo basso per essere incluso in un blocco vuoto.
- Gli script di sblocco per ogni input devono essere validi rispetto agli script di blocco dell'output corrispondenti.

Queste condizioni possono essere viste in dettaglio nelle funzioni AcceptToMemoryPool, CheckTransaction e CheckInputs nel client di riferimento bitcoin. Si noti che le condizioni cambiano nel tempo, per affrontare nuovi tipi di attacchi denial-of-service o talvolta per allentare le regole in modo da includere più tipi di transazioni.

Verificando indipendentemente ogni transazione così come viene ricevuta e prima di propagarla, ogni nodo crea un pool di transazioni valide (ma non confermate) note come *transaction pool*, *memory pool* o *mempool*.

Nodi di Mining

Alcuni nodi della rete bitcoin sono nodi specializzati chiamati *miners*. In [ch01_intro_what_is_bitcoin] abbiamo introdotto Jing, uno studente di ingegneria informatica a

Shanghai, in Cina, che è un minatore bitcoin. Jing guadagna bitcoin eseguendo un "mining rig", che è un sistema computer-hardware specializzato progettato per estrarre bitcoin. L'hardware di mining specializzato di Jing è connesso a un server che esegue un intero nodo bitcoin. A differenza di Jing, alcuni minatori non hanno un nodo completo, come vedremo in Le Mining Pool. Come ogni altro nodo completo, il nodo di Jing riceve e propaga transazioni non confermate sulla rete bitcoin. Il nodo di Jing, tuttavia, aggrega anche queste transazioni in nuovi blocchi.

Il nodo di Jing è in attesa di ricevere nuovi blocchi, propagati sulla rete bitcoin, come fanno tutti i nodi. Tuttavia, l'arrivo di un nuovo blocco ha un significato speciale per un nodo di data mining. La competizione tra i minatori termina efficacemente con la propagazione di un nuovo blocco che funge da annuncio di un vincitore. Per i minatori, ricevere un nuovo blocco significa che qualcun altro ha vinto la competizione e che i riceventi hanno perso. Tuttavia, la fine di un round di una competizione è anche l'inizio del prossimo round. Il nuovo blocco non è solo una bandiera a scacchi, che segna la fine della corsa; è anche la pistola di partenza nella corsa per il prossimo blocco.

Aggregare le transazioni in blocchi

Dopo aver convalidato le transazioni, un nodo bitcoin le aggiungerà al *memory pool* o *transaction pool*, dove le transazioni attendono fino a quando non possono essere incluse (minate) in un blocco. Il nodo di Jing raccoglie, convalida ed inoltra nuove transazioni proprio come qualsiasi altro nodo. A differenza degli altri nodi, tuttavia, il nodo di Jing aggregherà queste transazioni in un *candidate block* (ndt blocco candidato).

Seguiamo i blocchi che sono stati creati da quando Alice ha comprato un caffè dal Bar di Bob (vedi [cup_of_coffee]). La transazione di Alice è stata inclusa nel blocco 277.316. Per dimostrare alcuni concetti propri di questo capitolo, assumiamo che quel blocco sia stato minato dal sistema di mining di Jing e abbia seguito la transazione di Alice fino a quando questa non sia divenuta parte di questo nuovo blocco.

Il nodo di mining di Jing mantiene una copia locale della blockchain, la lista di tutti i blocchi creati dall'inizio del sistema bitcoin nel 2009. Quando Alice compra la tazza di caffè, il nodo di Jing ha assemblato una catena fino al blocco 277.314. Il nodo di Jing è in attesa di ricevere le transazioni, cercando di estrarre un nuovo blocco e anche attendendo l'arrivo di eventuali blocchi scoperti da altri nodi. Poiché il nodo di Jing è un nodo di mining, riceve il blocco 277.315 attraverso la rete bitcoin. L'arrivo di questo blocco indica la fine della competizione per il blocco 277.315 e l'inizio della competizione per creare il blocco 277.316.

Durante i precedenti 10 minuti, mentre il nodo di Jing stava cercando una soluzione per il blocco 277.315, stava anche raccogliendo le transazioni in preparazione per il prossimo blocco. Ormai ha raccolto alcune centinaia di transazioni nel pool di memoria. Dopo aver ricevuto il blocco 277.315 e averlo convalidato, il nodo di Jing controllerà anche tutte le transazioni nel pool di memoria e rimuoverà quelle che erano incluse nel blocco 277.315. Qualsiasi transazione rimanga nel pool di memoria non è confermata e attende di essere registrata in un nuovo blocco.

Il nodo di Jing costruisce subito un nuovo blocco vuoto, un candidato per il blocco 277.316. Questo blocco è detto blocco candidato perché non è ancora un blocco valido, in quanto non contiene una valida proof of work. Il blocco diventa valido solo se il miner riesce a trovare una soluzione per l'algoritmo di proof-of-work.

Età della Transazione, Fee e Priorità

Per costruire il blocco candidato, il nodo bitcoin di Jing seleziona le transazioni dal pool di memoria applicando una metrica di priorità a ciascuna transazione e aggiungendo prima le transazioni con priorità più alta . Le transazioni sono prioritarie in base all'età dell'UTXO che viene speso nei loro input, consentendo di privilegiare gli input vecchi e di alto valore su input nuovi e più piccoli. Le transazioni con priorità possono essere inviate senza commissioni, se c'è abbastanza spazio nel blocco.

La priorità di una transazione è calcolata come la somma del valore e dell'età degli input divisi per la dimensione totale della transazione:

```
Priority = Sum (Value of input * Input Age) / Transaction Size --- Priorità = Somma di
(Valore dell'Input * Età dell'Input) / Dimensione della Transazione
```

In questa equazione, il valore di un input è misurato nell'unità base, il satoshi (1/100mo di bitcoin). L'età di una UTXO è il numero di blocchi che sono "trascorsi" da quando l'UTXO è stata registrata nella blockchain, misurando il numero di blocchi di "profondità" dentro la blockchain in cui si trova. La dimensione della transazione è misurata in byte.

Affinché una transazione sia considerata ad "alta priorità", la sua priorità deve essere maggiore di 57.600.000, che corrisponde a un bitcoin (100m satoshis), con un giorno di età (144 blocchi), in una transazione di 250 byte di dimensione totale:

```
High Priority > 100,000,000 satoshi * 144 blocchi / 250 byte = 57,600,000
```

I primi 50 kilobyte di spazio di transazione in un blocco sono riservati alle transazioni con priorità elevata. Il nodo di Jing riempirà i primi 50 kilobyte, dando la priorità alle transazioni con la priorità più alta, indipendentemente dal costo. Ciò consente di elaborare transazioni ad alta priorità anche se comportano commissioni zero.

Il nodo di data mining di Jing riempie quindi il resto del blocco fino alla dimensione massima del blocco (MAX_BLOCK_SIZE nel codice), con transazioni che prevedono almeno la commissione minima, dando la priorità a quelle con la commissione più alta per kilobyte della transazione.

Se c'è uno spazio rimanente nel blocco, il nodo di mining di Jing sceglie di riempirlo con transazioni senza fee. Alcuni miner scelgono di effettuare mining di transazioni senza fee sulla base del minimo sforzo. Altri miner potrebbero scegliere di ignorare le transazioni senza fee.

Tutte le transazioni lasciate nel pool di memoria, dopo il riempimento del blocco, rimarranno nel pool per essere incluse nel blocco successivo. Man mano che le transazioni rimangono nel pool di memoria, i loro ingressi "invecchiano", mentre l'UTXO che spendono si inseriscono più in profondità nella blockchain con nuovi blocchi aggiunti in cima. Poiché la priorità di una transazione dipende dall'età dei suoi input, le transazioni che rimangono nel pool invecchiano e quindi aumentano in priorità. Alla fine una transazione senza commissioni potrebbe raggiungere una priorità sufficientemente alta da essere inclusa nel blocco gratuitamente.

Le transazioni Bitcoin non hanno un timeout di scadenza. Una transazione valida ora sarà valida

per sempre. Tuttavia, se una transazione viene propagata attraverso la rete solo una volta, persisterà solo fino a quando viene mantenuta in un pool di memoria del nodo di mining. Quando un nodo di mining viene riavviato, il suo pool di memoria viene cancellato, poiché è una forma di memoria temporanea non persistente. Sebbene una transazione valida possa essere stata propagata attraverso la rete, se non viene eseguita potrebbe eventualmente non risiedere nel pool di memoria di alcun minatore. Si prevede che il software Wallet ritrasmetterà tali transazioni o le ricostruirà con commissioni più elevate se non vengono eseguite correttamente entro un ragionevole lasso di tempo.

Quando il nodo di Jing aggrega tutte le transazioni dal pool di memoria, il nuovo blocco candidato ha 418 transazioni con commissioni di transazione totali pari a 0,09094928 bitcoin. Puoi vedere questo blocco nella blockchain usando l'interfaccia della riga di comando del client Bitcoin Core, come mostrato in Block 277,316.

```
$ bitcoin-cli getblockhash 277316
000000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4

$ bitcoin-cli getblock
000000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4
```

Example 3. Block 277,316

```
{
    "hash" : "0000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",
    "confirmations" : 35561,
    "size" : 218629,
    "height" : 277316,
    "version" : 2,
    "merkleroot" :
"c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e",
    "tx" : [
        "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
        "b268b45c59b39d759614757718b9918caf0ba9d97c56f3b91956ff877c503fbe",

        ... 417 more transactions ...

       ],
    "time" : 1388185914,
    "nonce" : 924591752,
    "bits" : "1903a30c",
    "difficulty" : 1180923195.25802612,
    "chainwork" :
"0000000000000000000000000000000000000000000000934695e92aaf53afa1a",
    "previousblockhash" :
"0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569",
    "nextblockhash" :
"000000000000000010236c269dd6ed714dd5db39d36b33959079d78dfd431ba7"
}
```

La transazione generatrice

La prima transazione aggiunta al blocco è una transazione speciale, denominata *generation transaction* o *coinbase transaction*. Questa transazione è costruita dal nodo di Jing ed è la sua ricompensa per lo sforzo di mining. Il nodo di Jing crea la transazione di generazione come pagamento sul proprio portafoglio: "Pagare Jing all'indirizzo inviando 25.09094928 bitcoin." L'ammontare totale della ricompensa che Jing raccoglie per il mining di un blocco è la somma della ricompensa di coinbase (25 nuovi bitcoin) e le commissioni di transazione (0.09094928) la somma di tutte le transazioni incluse nel blocco come mostrato in Transazione generativa:

```
$ bitcoin-cli getrawtransaction
d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f 1
```

Example 4. Transazione generativa


```
{
    "hex" :
"01000000010000000000000000000000000000000000000000000000000000000000000000ffffffff
f0f03443b0403858402062f503253482ffffffff0110c08d9500000000232102aa970c592640d19de
03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac00000000",
    "txid" : "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
    "version" : 1,
    "locktime" : 0,
    "vin" : [
        {
            "coinbase" : "03443b0403858402062f503253482f",
            "sequence" : 4294967295
        }
    ],
    "vout" : [
        {
            "value" : 25.09094928,
            "n" : 0,
            "scriptPubKey" : {
                "asm" :
"02aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21OP_CHECKSIG",
                "hex" :
"2102aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac",
                "reqSigs" : 1,
                "type" : "pubkey",
                "addresses" : [
                    "1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N"
                ]
            }
        }
    ],
    "blockhash" :
"000000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",
    "confirmations" : 35566,
    "time" : 1388185914,
    "blocktime" : 1388185914
}
```


A differenza delle normali transazioni, la transazione di generazione non consuma (spende) UTXO come input. Invece, ha solo un input, chiamato *coinbase*, che crea bitcoin dal nulla. La transazione di generazione ha un output, pagabile all'indirizzo bitcoin del minatore stesso. L'output della transazione di generazione invia il valore di 25.09094928 bitcoin all'indirizzo bitcoin del minatore, in questo caso 1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N.

Ricompensa Coinbase e Fee

Per costruire la transazione di generazione, il nodo di Jing calcola innanzitutto l'ammontare totale delle commissioni di transazione aggiungendo tutti gli input e gli output delle 418 transazioni che sono state aggiunte al blocco. Le tariffe sono calcolate come:

```
Fee Totali = Somma(Input) - Somma(Output)
```

Nel blocco 277.316, le fee di transazione totali sono di 0.09094928 bitcoin.

Successivamente, il nodo di Jing calcola la corretta ricompensa per il nuovo blocco. La ricompensa è calcolata tramite la block height, che inizia a 50 bitcoin per blocco ed è ridotta della metà ogni 210.000 blocchi. Visto che questo blocco è alla height 277.316, la ricompensa corretta è di 25 bitcoin.

Il calcolo si può vedere nella funzione GetBlockSubsidy nel client Bitcoin Core, come mostrato in Calculating the block reward — Function GetBlockSubsidy, Bitcoin Core Client, main.cpp.

Example 5. Calculating the block reward — Function GetBlockSubsidy, Bitcoin Core Client, main.cpp

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)
{
    int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;
    // Forza la ricompensa del blocco a zero quando il right shift è undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // La ricompensa è dimezzata ogni 210.000 blocchi, questo occorre
approssimativamente ogni 4 anni.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Il sussidio iniziale è calcolato in satoshi moltiplicando 50 con la costante COIN (100,000,000 satoshis). Questo imposta la ricompensa iniziale (nSubsidy) a 5 miliardi di satoshi.

Successivamente, la funzione calcola il numero di halvings che si sono verificati dividendo l'altezza del blocco corrente per l'intervallo di dimezzamento (SubsidyHalvingInterval). Nel caso del blocco 277.316, con un intervallo di dimezzamento ogni 210.000 blocchi, il risultato è 1 dimezzamento.

Il numero massimo di dimezzamenti consentito è 64, dopo questo il codice impone una ricompensa di zero (ritorna solo le fee) se il 64esimo dimezzamento viene superato.

Successivamente, la funzione utilizza l'operatorebinary-right-shift per dividere la ricompensa (nSubsidy) di due per ogni ciclo di dimezzamento. Nel caso del blocco 277.316, questo farebbe binary-right-shift sulla ricompensa di 5 miliardi di satoshi una volta (un dimezzamento) per ottenere 2,5 miliardi di satoshi, o 25 bitcoin. L'operatore binary-right-shift viene utilizzato perché è

più efficiente per la divisione per due rispetto alla divisione di un intero o in virgola mobile.

Infine, la ricompensa coinbase (nSubsidy) è aggiunta alle fee di transazione (nFees), e viene ritornata la somma.

La Struttura della Transazione Generativa

Con questi calcoli, il nodo di Jing costruisce la transazione generativa per pagarsi 25.09094928 bitcoin.

Come puoi vedere in Transazione generativa, la transazione di generazione ha un formato speciale. Invece di un input di transazione che specifica un UTXO precedente da spendere, ha un input "coinbase". Abbiamo esaminato gli input delle transazioni in [tx_in_structure]. Confrontiamo un input di transazione normale con un input di transazione generativa. La struttura di un "normale" input di transazione mostra la struttura di una transazione normale, mentre La struttura di una transazione di input generativa mostra la struttura dell'input della transazione di generativa.

Table 1. La struttura di un "normale" input di transazione

Dimensione	Campo	Descrizione
32 bytes	Hash della Transazione	Puntatore alla transazione contenente l'UTXO che andrà speso
4 byte	Indice dell'Output	Il numero dell'indice della UTXO da spendere, il primo è 0
1-9 bytes (VarInt)	Dimensione dell'Unlocking-Script	Lunghezza dell'Unlocking-Script in bytes, a seguire
Variable	Unlocking-Script	Uno script che completa le condizioni del locking script dell'UTXO.
4 bytes	Sequence Number	Attualmente disabilitato, funzione di Tx-replacement, impostato a 0xFFFFFFFF

Table 2. La struttura di una transazione di input generativa

Dimensione	Campo	Descrizione
32 byte	Hash di Transazione	Tutti i bit sono a zero: Non è un hash di referenza di una transazione
4 byte	Indice dell'Output	Tutti i bit sono uno: 0xFFFFFFFF
1-9 byte (VarInt)	Dimensione dei Dati Coinbase	Lunghezza dei dati coinbase, da 2 a 100 byte
Variabile	Dati Coinbase	Dati arbitrari usati per nonce extra e per tag di mining Nei blocchi v2, deve iniziare con la block height

Dimensione	Campo	Descrizione
4 byte	Numero di Sequenza	Impostato a 0xFFFFFFFF

In una transazione di generazione, i primi due campi sono impostati su valori che non rappresentano un riferimento UTXO. Invece di un "Transaction Hash", il primo campo viene riempito con 32 byte tutti impostati su zero. L' "Output Index" è riempito con 4 byte tutti impostati su 0xFF (255 decimale). Lo "Unlocking Script" è sostituito dai dati di coinbase, un campo dati arbitrario usato dai minatori.

Coinbase Data

Le transazioni di generazione non hanno un campo script di sblocco (a.k.a., scriptSig). Invece, questo campo viene sostituito dai dati di coinbase, che devono essere compresi tra 2 e 100 byte. Tranne che per i primi pochi byte, il resto dei dati di coinbase può essere usato dai minatori in qualsiasi modo essi vogliano; sono dati arbitrari.

Nel genesis block, ad esempio, Satoshi Nakamoto ha aggiunto il testo "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" nei dati di coinbase, utilizzandolo come prova della data e per trasmettere un messaggio. Attualmente, i minatori usano i dati della base di monete per includere valori extra nonce e stringhe che identificano la mining pool (ndt di appartenenza), come vedremo nelle seguenti sezioni.

I primi pochi byte della coinbase erano arbitrari, ma non è più così. Come da Bitcoin Improvement Proposal 34 (BIP0034), i blocchi della versione 2 (blocchi con il campo di versione impostato su 2) devono contenere l'indice di altezza del blocco come un'operazione di script "push" all'inizio del campo coinbase.

Nel blocco 277.316 vediamo che il coinbase (vedi Transazione generativa), che si trova nel campo "Unlocking Script" o scriptSig dell'ingresso della transazione, contiene il valore esadecimale 03443b0403858402062f503253482f. Decodifichiamo questo valore.

Il primo byte, 03, indica al motore di esecuzione dello script di inserire i tre byte successivi nello stack di script (vedi [tx_script_ops_table_pushdata]). I successivi tre byte, 0x443b04, sono l'altezza del blocco codificata nel formato little-endian (inizia dal byte meno significativo per finire col più significativo). Invertire l'ordine dei byte e il risultato è 0x043b44, che è 277.316 in decimale.

Le poche prossime cifre esadecimali (03858402062) sono usate per codificare un *nonce* extra (vedi La Soluzione del Nonce Extra), o valore casuale, usato per trovare una soluzione adatta alla proof of work.

La parte finale dei dati di coinbase (2f503253482f) è la stringa codificata ASCII /P2SH/, che indica che il nodo di mining che ha estratto questo blocco supporta miglioramento pay-to-script-hash (P2SH) definito in BIP0016. L'introduzione della funzionalità P2SH richiedeva un "voto" da parte dei minatori per approvare BIP0016 o BIP0017. Coloro che avallano l'implementazione di BIP0016 dovevano includere /P2SH/ nei loro dati di coinbase. Coloro che approvavano l'implementazione BIP0017 di P2SH dovevano includere la stringa p2sh/CHV nei loro dati di coinbase. Il BIP0016 è stato eletto come il vincitore e molti minatori hanno continuato a includere la stringa /P2SH/ nella loro coinbase per indicare il supporto per questa funzione.

Estrae i dati coinbase dal genesis block utilizza la libreria libbitcoin introdotta in [alt_libraries] per estrarre i dati della coinbase dal genesis block, visualizzando il messaggio di Satoshi. Si noti che la libreria libbitcoin contiene una copia statica del blocco genesis, quindi il codice di esempio può recuperare il blocco genesi direttamente dalla libreria.

Example 6. Estrae i dati coinbase dal genesis block

```
/*
  Display the genesis block message by Satoshi.
*/
#include <iostream>
#include <bitcoin/bitcoin.hpp>

int main()
{
    // Create genesis block.
    bc::block_type block = bc::genesis_block();
    // Genesis block contains a single coinbase transaction.
    assert(block.transactions.size() == 1);
    // Get first transaction in block (coinbase).
    const bc::transaction_type& coinbase_tx = block.transactions[0];
    // Coinbase tx has a single input.
    assert(coinbase_tx.inputs.size() == 1);
    const bc::transaction_input_type& coinbase_input = coinbase_tx.inputs[0];
    // Convert the input script to its raw format.
    const bc::data_chunk& raw_message = save_script(coinbase_input.script);
    // Convert this to an std::string.
    std::string message;
    message.resize(raw_message.size());
    std::copy(raw_message.begin(), raw_message.end(), message.begin());
    // Display the genesis block message.
    std::cout << message << std::endl;
    return 0;
}
```

Compiliamo il codice con il compilatore GNU C ++ ed eseguiamo l'eseguibile risultante, come mostrato in Compilando e lanciando il codice di esempio satoshi-words.

Example 7. Compilando e lanciando il codice di esempio satoshi-words

```
$ # Compila il codice
$  g++ -o satoshi-words satoshi-words.cpp $(pkg-config --cflags --libs libbitcoin)
$ # Esegue l'eseguibile
$ ./satoshi-words
^D    <GS>^A^DEThe Times 03/Jan/2009 Chancellor on brink of second bailout for
banks --- Il New York Times - 3 Gennaio 2009 - Il Cancelliere sull'orlo del
secondo salvataggio per le banche
```

Costruendo l'Header del Blocco

Per costruire il block header, il nodo di mining deve completare sei campi, come mostrato in La struttura di un block header.

Table 3. La struttura di un block header

Dimensione	Campo	Descrizione
4 byte	Versione	Un numero di versione per tracciare upgrade al software e/o al protocollo
32 byte	Hash del Blocco Precedente	Un riferimento all'hash del blocco precedente (genitore) nella chain
32 byte	Merkle Root	Un'hash della radice del merkle tree delle transazioni di questo blocco
4 byte	Timestamp	Il tempo approssimato della creazione del blocco corrente (secondi dalla Unix Epoch)
4 bytes	Target di Difficoltà (Difficulty Target)	Il target di difficoltà dell'algoritmo di proof-of-work per questo blocco
4 byte	Nonce	Un contatore utilizzato per l'algoritmo di proof-of-work

Nel tempo nel quale è stato effettuato mining sul blocco 277.316, il numero di versione che descrive la struttura del blocco è 2, che è codificato nel formato little-endian in 4 byte come 0x02000000.

Successivamente, il nodo di mining deve aggiungere il "Previous Block Hash" (ndt l'hash del blocco precedente). Questo è l'hash dell'header del blocco 277.315, il blocco precedente ricevuto dalla rete, che il nodo di Jing ha accettato e selezionato come genitore del blocco candidato 277.316. L'hash dell'intestazione del blocco per il blocco 277.315 è:

```
000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569
```

Il prossimo passo è quello di riepilogare tutte le transazioni con un merkle tree, in modo da aggiungere la radice del merkle tree, all'intestazione del blocco. La transazione di generazione è elencata come prima transazione nel blocco. Quindi, dopo di esso vengono aggiunte altre 418 transazioni, per un totale di 419 transazioni nel blocco. Come abbiamo visto nel [merkle_trees], ci deve essere un numero pari di nodi "foglia" nell'albero, quindi l'ultima transazione viene duplicata, creando 420 nodi, ciascuno contenente l'hash di una transazione. Gli hash della transazione vengono quindi combinati, in coppie, creando ogni livello dell'albero, fino a quando tutte le transazioni sono riepilogate in un nodo nella "radice" dell'albero. La radice del merkle tree riassume tutte le transazioni in un singolo valore di 32 byte, che puoi vedere elencato come "merkle root" in Block 277,316, e qui:

```
c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e
```

Il nodo di data mining aggiungerà quindi un timestamp a 4 byte, codificato come timestamp "Epoch" di Unix, che si basa sul numero di secondi trascorsi dalla mezzanotte del 1 ° gennaio 1970 UTC/GMT. Il tempo 1388185914 è uguale a venerdì, 27 dicembre 2013, 23:11:54 UTC/GMT.

Il nodo quindi riempie l'obiettivo di difficoltà, che definisce la difficoltà richiesta di prova del lavoro per rendere questo un blocco valido. La difficoltà è memorizzata nel blocco come una metrica "difficulty bits", che è una codifica mantissa-exponent del bersaglio. La codifica ha un esponente da 1 byte, seguito da un coefficiente di mantissa di 3 byte. Ad esempio, nel blocco 277.316, il valore dei bit di difficoltà è 0x1903a30c. La prima parte 0x19 è un esponente esadecimale, mentre la parte successiva, 0x03a30c, è il coefficiente. Il concetto di un obiettivo di difficoltà è spiegato in Il Target di Difficoltà e Il Retargeting e la rappresentazione del "difficulty bits" è spiegata in Rappresentazione della Difficoltà.

Il campo finale è il nonce, che è inizializzato a zero.

Con tutti gli altri campi riempiti, l'intestazione del blocco è ora completa e può iniziare il processo di estrazione. L'obiettivo è ora di trovare un valore per il nonce che generi un hash di intestazione del blocco, inferiore all'obiettivo di difficoltà. Il nodo di mining dovrà testare miliardi o trilioni di valori nonce prima che venga trovato un nonce che soddisfi i requisiti.

Effettuando Mining sul Blocco

Ora che un blocco candidato è stato costruito dal nodo di Jing, è tempo che l'hardware di Jing "mini" il blocco per trovare una soluzione all'algoritmo di proof-of-work (ndt prova di lavoro) che rende valido il blocco. In questo libro abbiamo studiato le funzioni hash crittografiche utilizzate in vari aspetti del sistema bitcoin. La funzione hash SHA256 è la funzione utilizzata nel processo di mining di bitcoin.

In termini più semplici, il mining è la ripetizione del processo di hashing dell'header del blocco, modificando un parametro (ndt nonce), fino a quando l'hash risultante corrisponde ad un target specifico. Il risultato della funzione di hash non può essere determinato in anticipo, né può essere creato uno schema che produca un valore hash specifico. Questa caratteristica delle funzioni di hash significa che l'unico modo per produrre un risultato di hash corrispondente ad un obiettivo specifico è di provare ancora e ancora, modificando casualmente l'input fino a quando il risultato

dell'hash desiderato appare per caso.

Algoritmo di Proof-Of-Work

Un algoritmo hash prende un input di dati di lunghezza arbitraria e produce un risultato deterministico a lunghezza fissa, un'impronta digitale digitale dell'input. Per ogni input specifico, l'hash risultante sarà sempre lo stesso e può essere facilmente calcolato e verificato da chiunque implementa lo stesso algoritmo di hash. La caratteristica chiave di un algoritmo di hash crittografico è che è praticamente impossibile trovare due input diversi che producono la stessa impronta digitale. Come corollario, è anche praticamente impossibile selezionare un input in modo tale da produrre un'impronta digitale desiderata, oltre a provare input casuali.

Con SHA256, l'output è sempre lungo 256 bit, indipendentemente dalla dimensione dell'input. In Esempio di SHA256, utilizzeremo l'interprete Python per calcolare l'hash SHA256 della frase "I am Satoshi Nakamoto."

Example 8. Esempio di SHA256

```
$ python
```

```
Python 2.7.1
>>> import hashlib
>>> print hashlib.sha256("I am Satoshi Nakamoto").hexdigest()
5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e
```

Esempio di SHA256 mostra il risultato del calcolo dell'hash di "I am Satoshi Nakamoto": 5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e. Questo numero a 256 bit è l' *hash* o *digest* della frase e dipende da ogni parte della frase. L'aggiunta di una singola lettera, un segno di punteggiatura o qualsiasi altro carattere produrrà un diverso hash.

Adesso, se cambiano la frase, dovremo aspettarci hash completamente differenti. Proviamolo aggiungendo un numero alla fine della nostra frase, usando il semplice script python SHA256 Uno script per generare molti hash iterando su di un nonce.

Example 9. SHA256 Uno script per generare molti hash iterando su di un nonce

```
# example of iterating a nonce in a hashing algorithm's input

import hashlib

text = "I am Satoshi Nakamoto"

# iterate nonce from 0 to 19
for nonce in xrange(20):

    # add the nonce to the end of the text
    input = text + str(nonce)

    # calculate the SHA-256 hash of the input (text+nonce)
    hash = hashlib.sha256(input).hexdigest()

    # show the input and hash result
    print input, '=>',  hash
```

Eseguendo questo, verranno prodotti gli hash di varie frasi, ottenute ognuna diversa dall'altra tramite l'aggiunta di un numero alla fine del testo. Incrementando il numero, possiamo ottenere hash differenti, come mostrato in Output SHA256 di uno script per generare molti hash iterando su di un nonce.

Example 10. Output SHA256 di uno script per generare molti hash iterando su di un nonce

```
$ python hash_example.py
```

```
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...
I am Satoshi Nakamoto6 => 4a2fd48e3be420d0d28e202360cfbaba...
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...
I am Satoshi Nakamoto14 => 27ead1ca85da66981fd9da01a8c6816...
I am Satoshi Nakamoto15 => 394809fb809c5f83ce97ab554a2812c...
I am Satoshi Nakamoto16 => 8fa4992219df33f50834465d3047429...
I am Satoshi Nakamoto17 => dca9b8b4f8d8e1521fa4eaa46f4f0cd...
I am Satoshi Nakamoto18 => 9989a401b2a3a318b01e9ca9a22b0f3...
I am Satoshi Nakamoto19 => cda56022ecb5b67b2bc93a2d764e75f...
```

Ogni frase produce un risultato hash completamente diverso. Sembrano completamente casuali, ma è possibile riprodurre i risultati esatti in questo esempio su qualsiasi computer con Python e vedere gli stessi hash esatti.

Il numero usato come variabile in tale scenario è chiamato *nonce*. Il nonce viene utilizzato per variare l'output di una funzione crittografica, in questo caso per variare l'impronta digitale SHA256 della frase.

Per creare una sfida a questo algoritmo, impostiamo un obiettivo arbitrario: trova una frase che produce un hash esadecimale che inizia con uno zero. Fortunatamente, questo non è difficile! Output SHA256 di uno script per generare molti hash iterando su di un nonce mostra che la frase "I am Satoshi Nakamoto13" produce l'hash 0ebc56d59a34f5082aaef3d66b37a661696c2b618e62432727216ba9531041a5, che corrisponde ai nostri criteri. Ci sono voluti 13 tentativi per trovarlo. In termini di probabilità, se l'output della funzione di hash è distribuito uniformemente ci aspetteremmo di trovare un risultato con uno 0 come prefisso esadecimale una volta ogni 16 hash (una su 16 cifre esadecimali da 0 a F). In termini numerici, ciò significa trovare un valore hash inferiore a 0x1000000000000000000000000000000000000000000000000000000000000000. Chiamiamo questa soglia il *target* e l'obiettivo è trovare un hash che sia numericamente più piccolo rispetto al target_. Se riduciamo il bersaglio, il compito di trovare un hash che è inferiore all'obiettivo diventa sempre più difficile.

Per fare una semplice analogia, immagina un gioco in cui i giocatori lanciano ripetutamente un paio di dadi, cercando di fare un numero inferiore ad un target specificato. Nel primo round, l'obiettivo è 12. A meno che non si lanci doppio-sei, si vince. Nel turno successivo l'obiettivo è 11. I giocatori devono fare 10 o meno per vincere, ancora una volta un compito facile. Diciamo alcuni round dopo che l'obiettivo è sceso a 5. Ora, più della metà dei tiri di dadi si sommano a più di 5 e quindi non validi. Ci vogliono più tiri per vincere, il numero di tiri cresce in modo esponenziale, più basso diventa il target. Alla fine, quando il bersaglio è 2 (il minimo possibile), solo un tiro su ogni 36, o il 2% di loro, produrrà un risultato vincente.

In Output SHA256 di uno script per generare molti hash iterando su di un nonce, il "nonce" vincente è 13 e questo risultato può essere confermato da chiunque in modo indipendente. Chiunque può aggiungere il numero 13 come suffisso alla frase "I am Satoshi Nakamoto" e calcolare l'hash, verificando che sia inferiore all'obiettivo. Il risultato positivo è anche la prova del lavoro, perché dimostra che abbiamo fatto il lavoro per trovare quel nonce. Mentre ci vuole solo un calcolo hash per verificare, ci sono voluti 13 calcoli hash per trovare un nonce che ha funzionato. Se avessimo un obiettivo inferiore (difficoltà più alta) ci vorrebbero molti più calcoli hash per trovare un nonce adatto, ma solo un calcolo hash per chiunque da verificare. Inoltre, conoscendo l'obiettivo, chiunque può stimare la difficoltà usando le statistiche e quindi sapere quanto lavoro è stato necessario per trovare un tale nonce.

La dimostrazione del lavoro di Bitcoin è molto simile alla sfida mostrata in Output SHA256 di uno script per generare molti hash iterando su di un nonce. Il minatore costruisce un blocco candidato pieno di transazioni. Successivamente, il minatore calcola l'hash dell'intestazione di questo blocco e vede se è più piccolo dell'attuale *target*. Se l'hash non è inferiore all'obiettivo, il minatore modificherà il nonce (di solito solo incrementandolo di uno) e riprova. Alla difficoltà attuale nella rete bitcoin, i minatori devono provare quadrilioni di volte prima di trovare un nonce che si traduce in un hash dell'intestazione di blocco abbastanza basso.

Una versione molto semplificata dell'algoritmo di proof-of-lavoro è implementata in Python in Implementazione proof-of-work semplificata.

Example 11. Implementazione proof-of-work semplificata

```
#!/usr/bin/env python
# example of proof-of-work algorithm

import hashlib
import time

max_nonce = 2 ** 32 # 4 billion

def proof_of_work(header, difficulty_bits):

    # calculate the difficulty target
    target = 2 ** (256-difficulty_bits)

    for nonce in xrange(max_nonce):
        hash_result = hashlib.sha256(str(header)+str(nonce)).hexdigest()
```

```
        # check if this is a valid result, below the target
        if long(hash_result, 16) < target:
            print "Success with nonce %d" % nonce
            print "Hash is %s" % hash_result
            return (hash_result,nonce)

    print "Failed after %d (max_nonce) tries" % nonce
    return nonce


if __name__ == '__main__':

    nonce = 0
    hash_result = ''

    # difficulty from 0 to 31 bits
    for difficulty_bits in xrange(32):

        difficulty = 2 ** difficulty_bits
        print "Difficulty: %ld (%d bits)" % (difficulty, difficulty_bits)

        print "Starting search..."

        # checkpoint the current time
        start_time = time.time()

        # make a new block which includes the hash from the previous block
        # we fake a block of transactions - just a string
        new_block = 'test block with transactions' + hash_result

        # find a valid nonce for the new block
        (hash_result, nonce) = proof_of_work(new_block, difficulty_bits)

        # checkpoint how long it took to find a result
        end_time = time.time()

        elapsed_time = end_time - start_time
        print "Elapsed Time: %.4f seconds" % elapsed_time

        if elapsed_time > 0:

            # estimate the hashes per second
            hash_power = float(long(nonce)/elapsed_time)
            print "Hashing Power: %ld hashes per second" % hash_power
```

Eseguendo questo codice, puoi scegliere la difficoltà (in bit, quanti dei primi bit devono avere valore zero) e vedere quanto tempo impiega il tuo computer per trovare una soluzione. In Eseguendo l'esempio di proof of work a varie difficoltà, puoi vedere come lavora su un laptop di fascia media.

Example 12. Eseguendo l'esempio di proof of work a varie difficoltà

```
$ python proof-of-work-example.py*
```

```
Difficulty: 1 (0 bits) --- Difficoltà: 1 (0 bits)

[...]

Difficulty: 8 (3 bits) --- Difficoltà: 8 (3 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 9 --- Trovato con nonce 9
L'hash è 1c1c105e65b47142f028a8f93ddf3dabb9260491bc64474738133ce5256cb3c1
Tempo Trascorso: 0.0004 secondi
Hashing Power: 25065 hashes per second
Difficulty: 16 (4 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 25
Hash is 0f7becfd3bcd1a82e06663c97176add89e7cae0268de46f94e7e11bc3863e148
Elapsed Time: 0.0005 seconds
Hashing Power: 52507 hashes per second
Difficulty: 32 (5 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 36
Hash is 029ae6e5004302a120630adcbb808452346ab1cf0b94c5189ba8bac1d47e7903
Elapsed Time: 0.0006 seconds
Hashing Power: 58164 hash per secondo

[...]

Difficulty: 4194304 (22 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 1759164
Hash is 0000008bb8f0e731f0496b8e530da984e85fb3cd2bd81882fe8ba3610b6cefc3
Elapsed Time: 13.3201 seconds
Hashing Power: 132068 hashes per second
Difficulty: 8388608 (23 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 14214729
Hash is 000001408cf12dbd20fcba6372a223e098d58786c6ff93488a9f74f5df4df0a3
Tempo Trascorso: 110.1507 secondi
Hashing Power: 129048 hashes per second
Difficulty: 16777216 (24 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 24586379
Hash is 0000002c3d6b370fccd699708d1b7cb4a94388595171366b944d68b2acce8b95
Elapsed Time: 195.2991 seconds
Hashing Power: 125890 hashes per second

[...]
```

```
Difficulty: 67108864 (26 bits)
Starting search... --- Inizio la ricerca...
Success with nonce 84561291
Hash is 0000001f0ea21e676b6dde5ad429b9d131a9f2b000802ab2f169cbca22b1e21a
Elapsed Time: 665.0949 seconds
Hashing Power: 127141 hashes per second
```

Come puoi vedere, l'aumento della difficoltà di 1 bit provoca un aumento esponenziale del tempo necessario per trovare una soluzione. Se si pensa all'intero spazio di 256 bit, ogni volta che si limita un altro bit a zero, si riduce lo spazio di ricerca della metà. In Eseguendo l'esempio di proof of work a varie difficoltà, ci vogliono 84 milioni di tentativi di hash per trovare un nonce che produce un hash con 26 bit iniziali come zero. Anche a una velocità di oltre 120.000 hash al secondo, richiede ancora 10 minuti su un laptop consumer per trovare questa soluzione.

Al momento della scrittura, la rete sta tentando di trovare un blocco il cui intestazione hash è inferiore a 000000000000004c296e6376db3a241271f43fd3f5de7ba18986e517a243baa7. Come puoi vedere, all'inizio di tale hash ci sono molti zeri, il che significa che l'intervallo accettabile di hash è molto più piccolo, quindi è più difficile trovare un hash valido. Ci vorranno in media più di 150 quadrilioni di calcoli di hash al secondo per consentire alla rete di scoprire il prossimo blocco. Sembra un compito impossibile, ma fortunatamente la rete sta portando a 100 petahashes al secondo (PH/sec) di potenza di elaborazione da sopportare, che sarà in grado di trovare un blocco in media in circa 10 minuti.

Rappresentazione della Difficoltà

In Block 277,316, abbiamo visto che il blocco contiene l'obiettivo di difficoltà, in una notazione chiamata "difficulty bits" o semplicemente "bits", che nel blocco 277.316 ha il valore di 0x1903a30c. Questa notazione esprime l'obiettivo di difficoltà come formato coefficiente/esponente, con le prime due cifre esadecimali per l'esponente e le successive sei cifre esadecimali come coefficiente. In questo blocco, quindi, l'esponente è 0x19 e il coefficiente è 0x03a30c.

La formula per calcolare il target di difficoltà da questa rappresentazione è:

```
target = coefficient * 2^(8 * (exponent – 3))
```

Usando quella formula, e i bit di difficoltà con valore 0x1903a30c, otteniamo:

```
target = 0x03a30c * 2^(0x08 * (0x19 - 0x03))^

=> target = 0x03a30c * 2^(0x08 * 0x16)^

=> target = 0x03a30c * 2^0xB0^
```

il quale decimale è:

```
=> target = 238,348 * 2^176^

=> target =
22,829,202,948,393,929,850,749,706,076,701,368,331,072,452,018,388,575,715,328
```

ri-convertendolo in esadecimale:

```
=> target = 0x00000000000000003A30C00000000000000000000000000000000000000000
```

Ciò significa che un blocco valido per l'altezza 277.316 è uno che ha un hash di intestazione di blocco inferiore alla destinazione. In binario, quel numero avrebbe più di 60 bit impostati a zero. Con questo livello di difficoltà, un singolo minatore che elabora 1 trilione di hash al secondo (1 tera-hash al secondo o 1 TH/sec) troverà una soluzione solo una volta ogni 8.496 blocchi o una volta ogni 59 giorni, in media.

Il Target di Difficoltà e Il Retargeting

Come abbiamo visto, il target determina la difficoltà e quindi influenza il tempo necessario per trovare una soluzione da parte dell'algoritmo di proof of work. Questo porta alle domande ovvie: perché la difficoltà è regolabile, chi la regola e come?

I blocchi di Bitcoin vengono generati ogni 10 minuti, in media. Questo è il battito (ndt del cuore) del bitcoin e sostiene la frequenza dell'emissione di valuta e la velocità del regolamento delle transazioni. Deve rimanere costante non solo nel breve periodo, ma per un periodo di molti decenni. In questo periodo, si prevede che la potenza dei computer continuerà ad aumentare a un ritmo rapido. Inoltre, anche il numero di partecipanti nel settore del mining e i computer che usano cambierà costantemente. Per mantenere il tempo di generazione del blocco a 10 minuti, la difficoltà di estrazione deve essere regolata per tenere conto di questi cambiamenti. Infatti, la difficoltà è un parametro dinamico che verrà periodicamente regolato per raggiungere un target di blocco di 10 minuti. In termini semplici, l'obiettivo di difficoltà farà si che qualsiasi potenza di mining sarà disponibile, genererà un blocco ogni 10 minuti.

In che modo, quindi, tale adeguamento viene effettuato in una rete completamente decentralizzata? Il retargeting di difficoltà si verifica automaticamente e su ogni nodo indipendentemente. Ogni 2.016 blocchi, tutti i nodi ritarano la difficoltà proof of work. L'equazione per la difficoltà di retargeting misura il tempo impiegato per trovare gli ultimi 2.016 blocchi e li confronta con il tempo previsto di 20.160 minuti (due settimane in base al tempo di blocco desiderato di 10 minuti). Viene calcolato il rapporto tra il periodo di tempo effettivo e il periodo di tempo desiderato e viene effettuata una correzione corrispondente (in su o giù) alla difficoltà. In termini semplici: se la rete trova blocchi più velocemente di ogni 10 minuti, la difficoltà aumenta. Se la scoperta dei blocchi è più lenta del previsto, la difficoltà diminuisce.

L'equazione può essere riassunta in:

```
New Difficulty = Old Difficulty * (Tempo degli ultimi 2016 Blocchi / 20160 minuti)
```

Retargeting the proof-of-work difficulty — CalculateNextWorkRequired() in pow.cpp mostra il codice utilizzato nel client Bitcoin Core.

Example 13. Retargeting the proof-of-work difficulty — CalculateNextWorkRequired() in pow.cpp

```
// Step di aggiustamento del limite
 int64_t nActualTimespan = pindexLast->GetBlockTime() - nFirstBlockTime;
 LogPrintf("  nActualTimespan = %d  before bounds\n", nActualTimespan);
 if (nActualTimespan < params.nPowTargetTimespan/4)
     nActualTimespan = params.nPowTargetTimespan/4;
 if (nActualTimespan > params.nPowTargetTimespan*4)
     nActualTimespan = params.nPowTargetTimespan*4;

 // Retarget
 const arith_uint256 bnPowLimit = UintToArith256(params.powLimit);
 arith_uint256 bnNew;
 arith_uint256 bnOld;
 bnNew.SetCompact(pindexLast->nBits);
 bnOld = bnNew;
 bnNew *= nActualTimespan;
 bnNew /= params.nPowTargetTimespan;

 if (bnNew > bnPowLimit)
     bnNew = bnPowLimit;
```

NOTE Mentre la calibrazione della difficoltà avviene ogni 2.016 blocchi, a causa di un errore "off-by-one" nel client Bitcoin Core originale, si basa sul tempo totale dei precedenti 2.015 blocchi (non a 2.016 come dovrebbe essere), determinando un bias di retargeting verso una maggiore difficoltà dello 0,05%.

I parametri Intervallo (2.016 blocchi) e TargetTimespan (due settimane come 1.209.600 secondi) sono definiti in *chainparams.cpp*.

Per evitare la volatilità estrema nella difficoltà, la regolazione del retargeting deve essere inferiore a un fattore di quattro (4) per ciclo. Se la regolazione della difficoltà richiesta è maggiore di un fattore di quattro, sarà regolata al massimo e non di più. Qualsiasi ulteriore aggiustamento verrà effettuato nel prossimo periodo di retargeting, in quanto lo squilibrio persisterà nei successivi 2.016 blocchi. Pertanto, grandi discrepanze tra la potenza di hash e la difficoltà potrebbero richiedere diversi cicli da 2.016 blocchi per bilanciare.

TIP La difficoltà nel trovare un blocco bitcoin è approssivativamente '10 minuti di calcolo' per l'intero network, basato sul tempo che ci vuole per trovare i precedenti 2.016 blocchi, aggiustata ogni 2.016 blocchi.

Si noti che la difficoltà di destinazione è indipendente dal numero di transazioni o dal valore delle transazioni. Ciò significa che la quantità di potenza di hash, e quindi di energia spesa per proteggere bitcoin, è del tutto indipendente dal numero di transazioni. Bitcoin può scalare, ottenere

un'adozione più ampia e rimanere al sicuro senza alcun aumento della potenza di hashing dal livello attuale. L'aumento della potenza di hash rappresenta le forze del mercato quando nuovi minatori entrano nel mercato per competere per la ricompensa. Fintanto che un sufficiente potere di hashing è sotto il controllo dei minatori che agiscono onestamente alla ricerca della ricompensa, è sufficiente per prevenire attacchi di "takeover" e, quindi, è sufficiente per proteggere i bitcoin.

La difficoltà dell'obiettivo è strettamente correlata al costo dell'elettricità e al tasso di cambio del bitcoin rispetto alla valuta utilizzata per pagare l'elettricità. I sistemi di estrazione ad alte prestazioni sono quanto più efficienti possibile con l'attuale generazione di device al silicio, convertendo l'elettricità in calcolo dell'hashing alla massima velocità possibile. L'influenza primaria sul mercato dei miner è il prezzo di un chilowattora in bitcoin, perché questo determina la redditività del settore mining e quindi gli incentivi per entrare o uscire dal mercato del mining.

Effettuando Mining del Blocco con Successo

Come abbiamo visto prima, il nodo di Jing ha costruito un blocco candidato e lo ha preparato per il mining. Jing ha diverse piattaforme hardware mining circuiti integrati specifici dell'applicazione, dove centinaia di migliaia di circuiti integrati eseguono l'algoritmo SHA256 in parallelo a velocità incredibili. Queste macchine specializzate sono collegate al suo nodo di mining via USB. Successivamente, il nodo di mining in esecuzione sul desktop di Jing trasmette l'intestazione del blocco al suo hardware di data mining, che inizia a testare trilioni di nonce al secondo.

Quasi 11 minuti dopo l'avvio del mio blocco 277.316, una delle macchine di mining hardware trova una soluzione e la rimanda al nodo di data mining. Quando inserito nell'intestazione del blocco, il nonce 4.215.469.401 produce un hash di blocco di:

```
0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569
```

che è inferiore al target:

```
00000000000000003A30C00000000000000000000000000000000000000000000
```

Immediatamente, il nodo di mining di Jing trasmette il blocco a tutti i suoi pari. Essi ricevono, convalidano e quindi propagano il nuovo blocco. Mentre il blocco si diffonde attraverso la rete, ogni nodo lo aggiunge alla propria copia della blockchain, estendendolo a una nuova altezza di 277.316 blocchi. Mentre i nodi di mining ricevono e convalidano il blocco, abbandonano i loro sforzi per trovare un blocco alla stessa altezza e iniziano immediatamente a calcolare il prossimo blocco della catena.

Nella prossima sezione, osserveremo il processo usato da ogni nodo per validare un blocco e selezionare la catena più lunga, creando il consenso che forma la blockchain decentralizzata.

Validando un Nuovo Blocco

Il terzo passo nel meccanismo di consenso di bitcoin è la convalida indipendente di ogni nuovo blocco da parte di ogni nodo della rete. Mentre il blocco appena risolto si propaga attraverso la rete,

ciascun nodo esegue una serie di test per convalidarlo prima di propagarlo ai suoi pari. Ciò garantisce che solo i blocchi validi vengano propagati sulla rete. La convalida indipendente assicura anche che i minatori che agiscono in modo onesto ottengano i loro blocchi incorporati nella blockchain, guadagnando così la ricompensa. Quei minatori che agiscono disonestamente hanno i loro blocchi respinti e non solo perdono la ricompensa, ma anche sprecano lo sforzo speso per trovare una soluzione di proof of work, incorrendo così nel costo dell'elettricità senza compensazione.

Quando un nodo riceve un nuovo blocco, convaliderà il blocco controllandolo da una lunga lista di criteri che devono essere soddisfatti tutti; in caso contrario, il blocco viene rifiutato. Questi criteri possono essere visualizzati nel client Bitcoin Core nelle funzioni CheckBlock e CheckBlockHeader e include:

- La struttura del blocco è sintatticamente valida
- L'hash del block header è inferiore della difficoltà del target (impone la proof of work)
- Il timestamp del blocco è inferiore di due ore nel futuro (permettendo errori temporali) *La dimensione del blocco è entro i limiti accettati
- La prima transazione (e solo la prima) è una transazione di generazione coinbase
- Tutte le transazioni nel blocco sono valide usando la checklist di transazione introdotta in Verifica Indipendente delle Transazioni

La convalida indipendente di ogni nuovo blocco da parte di ogni nodo della rete garantisce che i minatori non possano imbrogliare. Nelle sezioni precedenti abbiamo visto come i minatori possono scrivere una transazione che assegna loro i nuovi bitcoin creati all'interno del blocco e rivendicare le commissioni di transazione. Perché i minatori non scrivono una transazione per un migliaio di bitcoin invece della ricompensa giusta? Perché ogni nodo convalida i blocchi secondo le stesse regole. Una transazione di coinbase non valida renderebbe l'intero blocco non valido, il che comporterebbe il rifiuto del blocco e, pertanto, tale transazione non diventerebbe mai parte del libro mastro. I minatori devono costruire un blocco perfetto, basato sulle regole condivise seguite da tutti i nodi, e collegarlo con una soluzione corretta alla prova di lavoro. Per fare ciò, spendono molta elettricità nel mining e, se imbrogliano, tutta l'elettricità e lo sforzo sono sprecati. Questo è il motivo per cui la convalida indipendente è una componente chiave del consenso decentralizzato.

Assemblando e Selezionando Catene di Blocchi

Il passo finale nel meccanismo di consenso decentralizzato di bitcoin è l'assemblaggio di blocchi in catene e la selezione della catena con la maggior prova di lavoro. Una volta che un nodo ha convalidato un nuovo blocco, tenterà quindi di assemblare una catena collegando il blocco alla blockchain esistente.

I nodi mantengono tre tipi di blocchi: quelli connessi alla blockchain principale, quelli che formano rami che partono dalla blockchain principale (chain secondarie), ed infine, blocchi che non hanno un genitore presente nelle chain conosciute (orfani). I blocchi non validi sono rifiutati non appena uno dei criteri di validazione fallisce e quindi non vengono inclusi in nessuna chain.

In qualsiasi momento, la "catena principale" è la catena di blocchi che avrà accumulato maggior difficoltà. Nella maggior parte dei casi questa è anche la catena con il maggior numero di blocchi, a

meno che non ci siano due catene di uguale lunghezza e una abbia più prove di lavoro. La catena principale avrà anche rami con blocchi che sono "fratelli" per i blocchi sulla catena principale. Questi blocchi sono validi ma non fanno parte della catena principale. Sono conservati per riferimento futuro, nel caso in cui una di queste catene sia estesa per superare la catena principale in difficoltà. Nella prossima sezione (I Fork della Blockchain), vedremo come si verificano le catene secondarie come risultato di un mining quasi simultaneo di blocchi alla stessa altezza.

Quando viene ricevuto un nuovo blocco, un nodo proverà a inserirlo nella blockchain esistente. Il nodo esaminerà il campo "hash del blocco precedente" del blocco, che è il riferimento al genitore del nuovo blocco. Quindi, il nodo tenterà di trovare quel genitore nella blockchain esistente. La maggior parte delle volte, il genitore sarà il "tip" (ndt ultimo blocco accodato) della catena principale, il che significa che questo nuovo blocco estende la catena principale. Ad esempio, il nuovo blocco 277.316 ha un riferimento all'hash del suo blocco padre 277.315. La maggior parte dei nodi che ricevono 277.316 avranno già il blocco 277.315 come punta della loro catena principale e quindi collegheranno il nuovo blocco ed estenderanno quella catena.

A volte, come vedremo in I Fork della Blockchain, il nuovo blocco estende una catena che non è la catena principale. In tal caso, il nodo assegnerà il nuovo blocco alla catena secondaria che estende e quindi confronterà la difficoltà della catena secondaria con la catena principale. Se la catena secondaria ha più difficoltà cumulative della catena principale, il nodo sarà *reconverge* sulla catena secondaria, il che significa che selezionerà la catena secondaria come la sua nuova catena principale, rendendo la vecchia catena principale una catena secondaria. Se il nodo è un minatore, ora costruirà un blocco che estende questa nuova catena più lunga.

Se viene ricevuto un blocco valido e nessun genitore viene trovato nelle catene esistenti, quel blocco viene considerato un "orfano". I blocchi orfani vengono salvati nel pool di blocchi orfani dove rimarranno fino alla ricezione del genitore. Una volta che il genitore è stato ricevuto e collegato nelle catene esistenti, l'orfano può essere estratto dal pool orfano e collegato al genitore, rendendolo parte di una catena. I blocchi orfani di solito si verificano quando due blocchi che sono stati estratti in un breve lasso di tempo sono ricevuti in ordine inverso (figlio prima del genitore).

Selezionando la catena con la maggior difficoltà, tutti i nodi raggiungono infine un consenso a livello di rete. Le discrepanze temporanee tra le catene sono risolte con l'aggiunta di ulteriori prove di lavoro, estendendo una delle possibili catene. I nodi di mining "votano" con la loro potenza di calcolo scegliendo quale catena estendere estraendo il blocco successivo. Quando estraggono un nuovo blocco ed estendono la catena, il nuovo blocco rappresenta il loro voto.

Nella prossima sezione daremo un'occhiata a come discrepanze tra catene concorrenti (fork) sono risolte dalla selezione indipendente della catena di difficolta piu lunga.

I Fork della Blockchain

Visto che la blockchain è una struttura dati decentralizzata, le diverse copie di essa non sono sempre identiche. I blocchi potrebbero arrivare ai vari nodi in momenti differenti, questo fa si che i nodi abbiano diverse prospettive della blockchain. Per risolvere questo problema, ogni nodo seleziona e cerca di estendere sempre la catena dei blocchi che rappresenta la maggiore proof-of-work, conosciuta anche come la longest chain (la catena più lunga) o la catena piu grande e con difficoltà cumulativa più alta. Sommando la difficoltà registrata in ogni blocco in una chain, un nodo può calcolare la quantità totale di proof-of-work che è stata spesa per creare quella chain.

Fino a quando tutti i nodi selezionano la catena con difficoltà cumulativa più lunga, la rete globale bitcoin eventualmente convergerà verso uno stato consistente. Le ramificazioni (fork) avvengono come inconsistenze temporanee tra le varie versioni della blockchain, che sono risolte da eventuali riconvergenze mano a mano che più blocchi sono aggiunti a uno dei rami.

Nei prossimi diagrammi, seguiamo l'andamento di un evento "fork" attraverso la rete. Il diagramma è una rappresentazione semplificata di bitcoin come rete globale. In realtà, la topologia della rete bitcoin non è organizzata geograficamente. Piuttosto, forma una rete a maglie di nodi interconnessi, che potrebbero essere localizzati geograficamente molto lontani l'uno dall'altro. La rappresentazione di una topologia geografica è una semplificazione utilizzata allo scopo di illustrare un fork. Nella vera rete di bitcoin, la "distanza" tra i nodi viene misurata in "salti" da un nodo all'altro, non sulla loro posizione fisica. A scopo illustrativo, i diversi blocchi sono mostrati come colori diversi, si diffondono attraverso la rete e colorano le connessioni che attraversano.

Nel primo diagramma (Visualizzazione di un'evento di fork di blockchain—prima del fork), il network ha una prospettiva unificata della blockchain, con il blocco blu come estremità della chain principale.

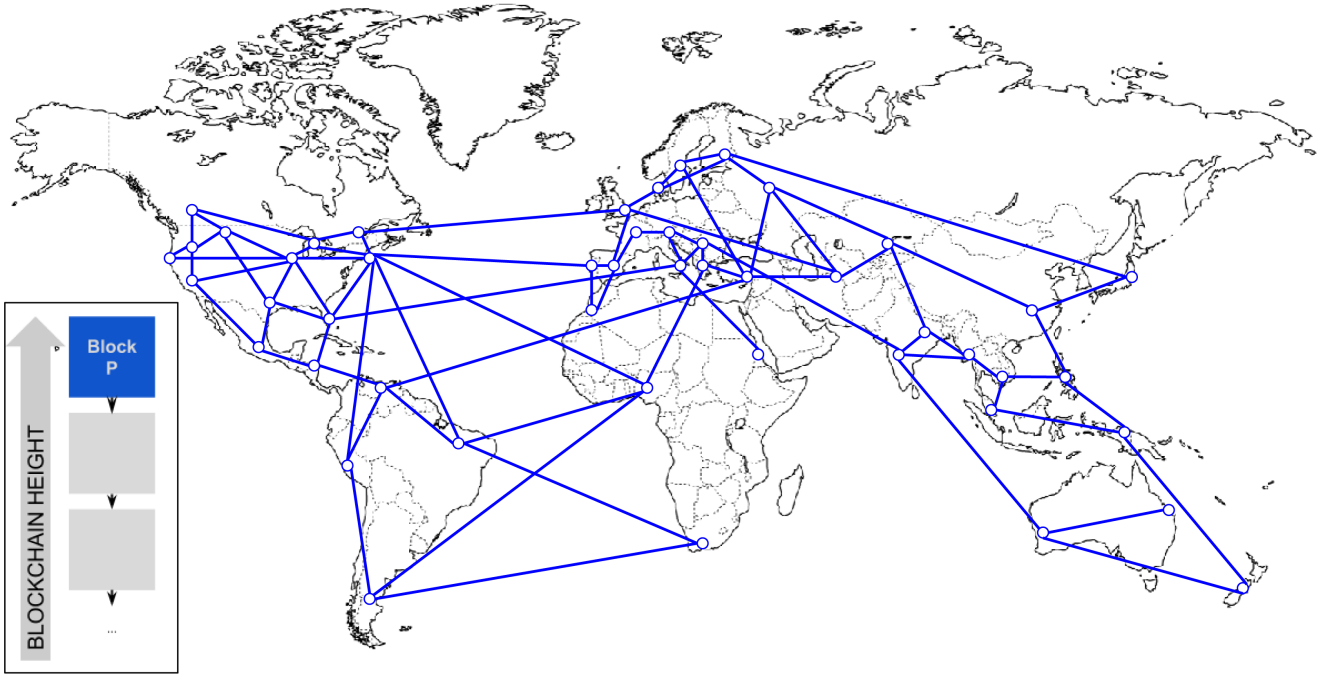

Figure 2. Visualizzazione di un'evento di fork di blockchain—prima del fork

Un "fork" si verifica ogni volta che ci sono due blocchi candidati in competizione per formare la blockchain più lunga. Ciò si verifica in condizioni normali ogni volta che due minatori risolvono l'algoritmo di prova di lavoro entro un breve periodo di tempo l'uno dall'altro. Quando entrambi i minatori scoprono una soluzione per i rispettivi blocchi candidati, trasmettono immediatamente il proprio blocco "vincente" ai loro vicini immediati che iniziano a propagare il blocco attraverso la rete. Ogni nodo che riceve un blocco valido lo incorporerà nella sua blockchain, estendendo la blockchain di un blocco. Se in seguito tale nodo vede un altro blocco candidato che estende lo stesso genitore, collega il secondo candidato su una catena secondaria. Di conseguenza, alcuni nodi "vedranno" prima un blocco candidato, mentre altri nodi vedranno emergere l'altro blocco candidato e due versioni concorrenti della blockchain.

In Visualizzazione di un'evento di fork di blockchain: due blocchi trovati simultaneamente, vediamo due minatori che estraggono due blocchi diversi quasi simultaneamente. Entrambi questi

blocchi sono figli del blocco blu, pensato per estendere la catena costruendo sopra il blocco blu. Per aiutarci a rintracciarlo, uno viene visualizzato come un blocco rosso proveniente dal Canada e l'altro è contrassegnato come un blocco verde originario dell'Australia.

Supponiamo, per esempio, che un minatore in Canada trovi una soluzione di prova del lavoro per un blocco "rosso" che estende la blockchain, costruendo sopra il blocco principale "blu". Quasi contemporaneamente, un minatore australiano che stava anche estendendo il blocco "blu" trova una soluzione per il blocco "verde", il suo blocco candidato. Ora, ci sono due possibili blocchi, uno che chiamiamo "rosso", originario del Canada, e uno che chiamiamo "verde", originario dell'Australia. Entrambi i blocchi sono validi, entrambi i blocchi contengono una soluzione valida per la prova di lavoro, ed entrambi i blocchi estendono lo stesso genitore. Entrambi i blocchi probabilmente contengono la maggior parte delle stesse transazioni, con solo forse alcune differenze nell'ordine delle transazioni.

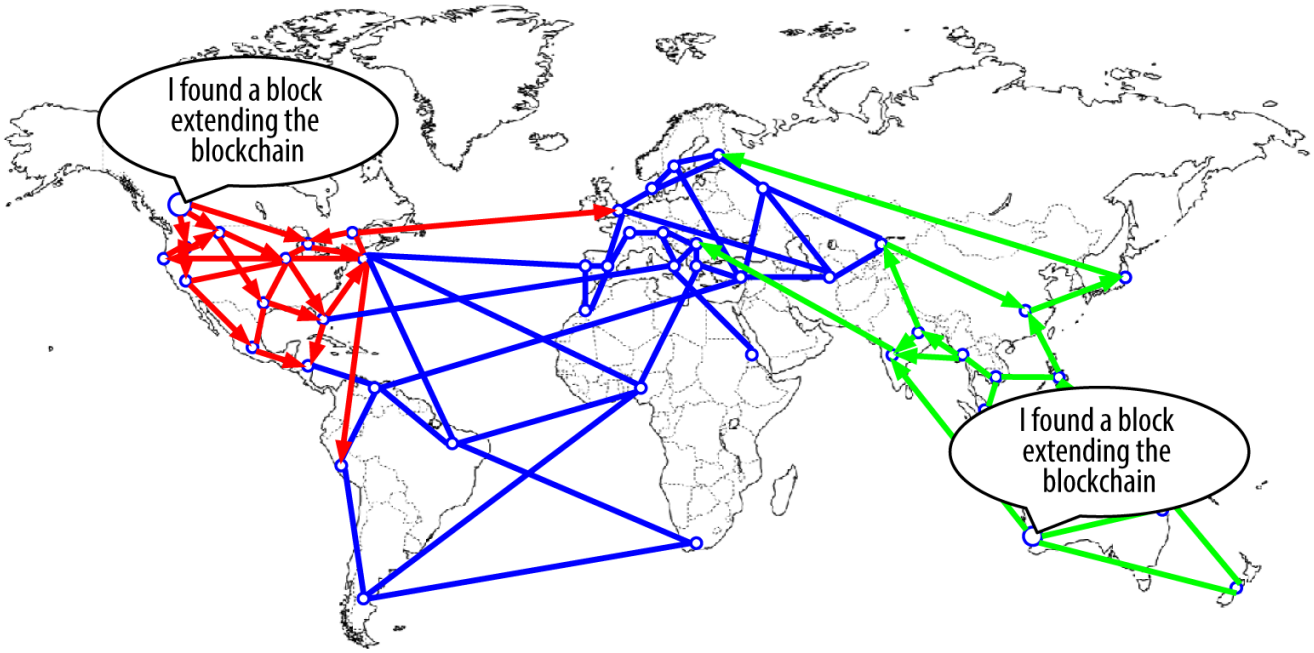

Figure 3. Visualizzazione di un'evento di fork di blockchain: due blocchi trovati simultaneamente

Mentre i due blocchi si propagano, alcuni nodi ricevono prima il blocco "rosso" e alcuni ricevono prima il blocco "verde". Come mostrato in Visualizzazione di un'evento di fork di blockchain: due blocchi propagati, dividendo il network, la rete si divide in due diversi punti di vista della blockchain, un lato con un blocco rosso, l'altro con un blocco verde.

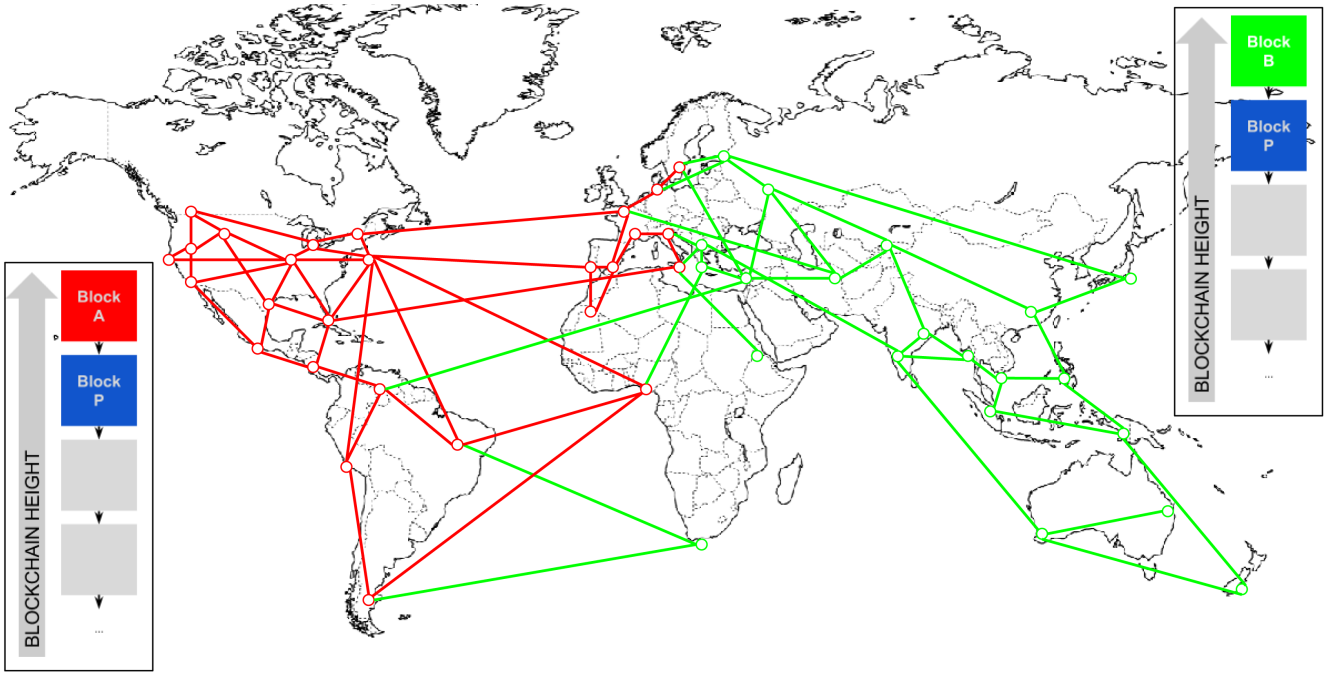

Figure 4. Visualizzazione di un'evento di fork di blockchain: due blocchi propagati, dividendo il network

Da quel momento, i nodi della rete bitcoin più vicini (topologicamente, non geograficamente) al nodo canadese sentiranno prima il blocco "rosso" e creeranno una nuova blockchain con una maggior difficoltà accumulata con "rosso" come ultimo blocco della catena (ad esempio, blu-rosso), ignorando il blocco candidato "verde" che arriva un po più tardi. Nel frattempo, i nodi più vicini al nodo australiano prenderanno quel blocco come vincitore e estenderanno la blockchain con "verde" come ultimo blocco (ad esempio, blu-verde), ignorando "rosso" quando arriverà qualche secondo dopo. Tutti i minatori che hanno visto "rosso" per primi costruiranno immediatamente i blocchi candidati che fanno riferimento a "rosso" come genitore e iniziano a provare a risolvere la prova di lavoro per questi blocchi candidati. I minatori che hanno accettato "verde" invece inizieranno a costruire in cima al "verde" e ad estendere quella catena.

I fork sono quasi sempre risolti in un blocco. Come parte della potenza di hashing della rete è dedicato alla costruzione di "rosso" come genitore, un'altra parte della potenza di hashing è incentrata sulla costruzione di "verde". Anche se il potere di hashing è diviso in modo quasi uniforme, è probabile che un gruppo di minatori troverà una soluzione e la propagherà prima che l'altro gruppo di minatori abbia trovato qualche soluzione. Diciamo, per esempio, che i minatori che si trovano in cima a "verde" trovano un nuovo blocco "rosa" che estende la catena (ad esempio, blu-verde-rosa). Propagano immediatamente questo nuovo blocco e l'intera rete lo vede come una soluzione valida come mostrato in Visualizzazione di un'evento di fork di blockchain: un nuovo blocco estende un fork.

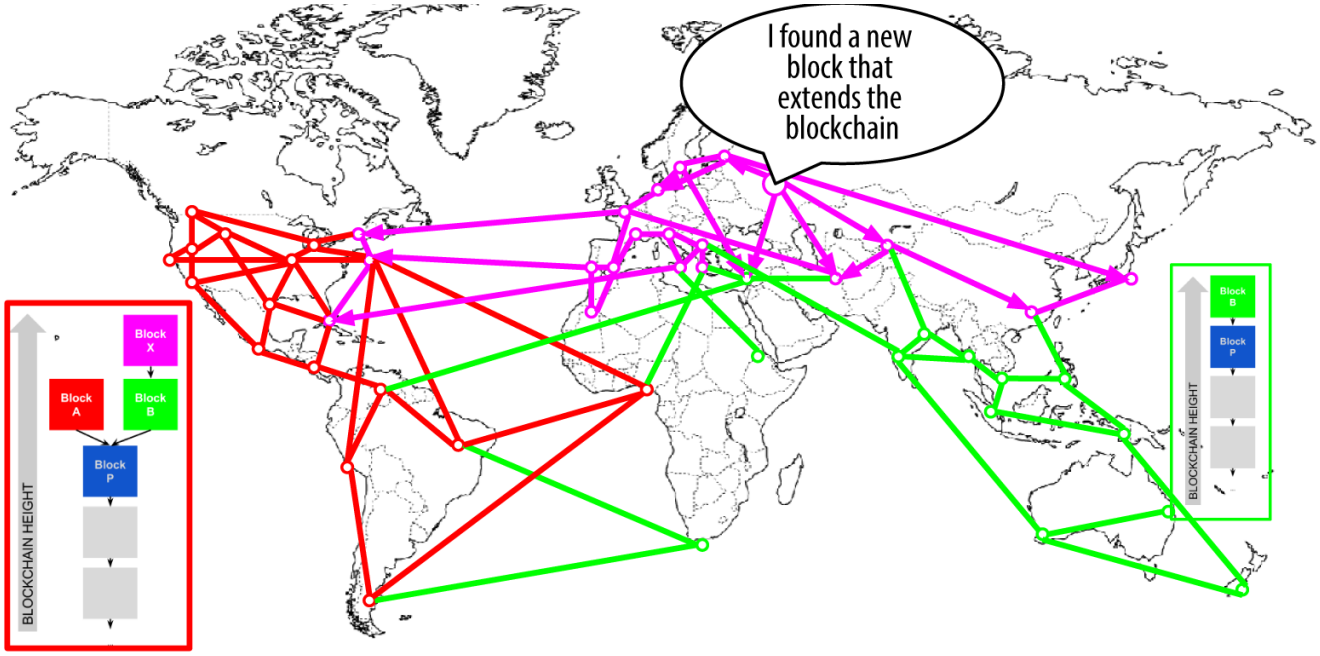

Figure 5. Visualizzazione di un'evento di fork di blockchain: un nuovo blocco estende un fork

Tutti i nodi che hanno scelto "verde" come vincitore nel round precedente estenderanno semplicemente un ulteriore blocco alla catena. I nodi che hanno scelto "rosso" come vincitore, tuttavia, vedranno ora due catene: blu-verde-rosa e blu-rosso. La catena blu-verde-rosa è ora più lunga (più difficoltà cumulativa) rispetto alla catena blu-rossa. Di conseguenza, questi nodi imposteranno la catena blu-verde-rosa come catena principale e trasformeranno la catena blu-rossa in catena secondaria, come mostrato in Visualizzazione di un'evento di fork di blockchain: il network riconverge su di una nuova longest chain. Questa è una riconversione a catena, perché quei nodi sono costretti a rivedere la loro visione della blockchain per incorporare la nuova evidenza di una catena più lunga. Tutti i miner che lavorano sull'estensione della catena blu-rossa ora interromperanno il lavoro perché il loro blocco candidato è un "orfano", poiché il suo genitore "rosso" non è più sulla catena più lunga. Le transazioni all'interno di "rosso" vengono nuovamente messe in coda per l'elaborazione nel blocco successivo, poiché tale blocco non si trova più nella catena principale. L'intera rete riconverge su una singola blockchain blu-verde-rosa, con "rosa" come ultimo blocco della catena. Tutti i minatori iniziano immediatamente a lavorare su blocchi candidati che fanno riferimento a "rosa" come genitore per estendere la catena blu-verde-rosa.

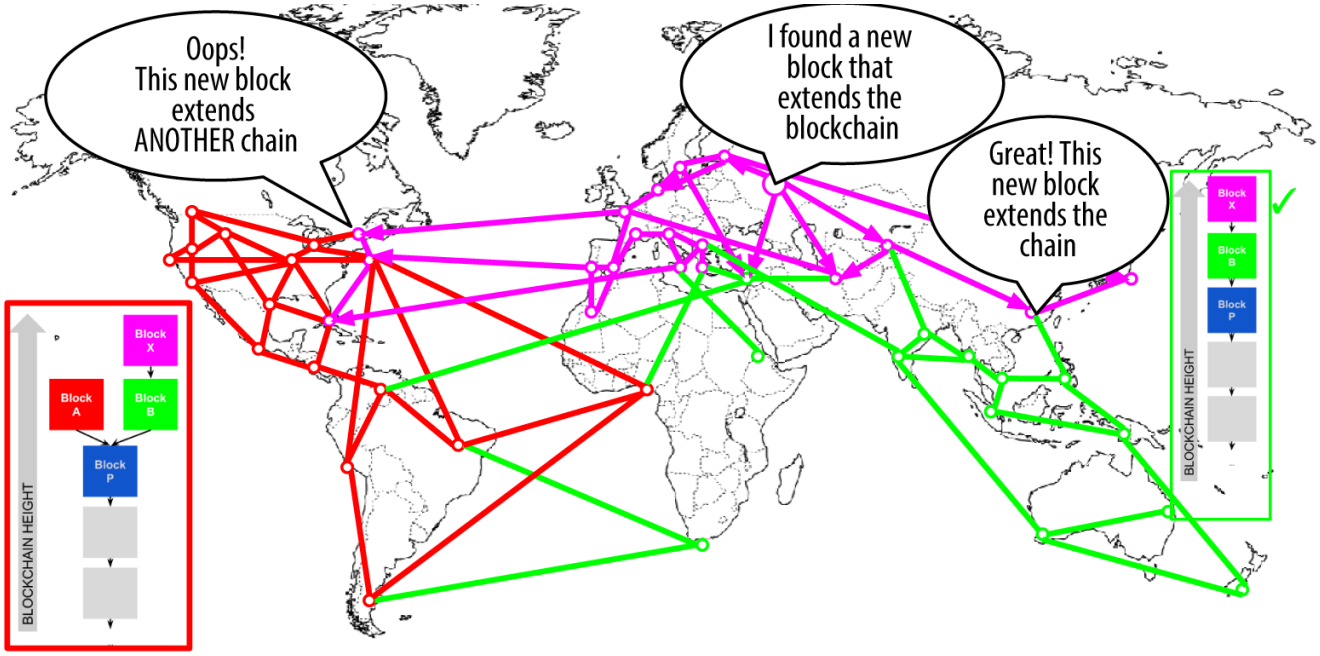

Figure 6. Visualizzazione di un'evento di fork di blockchain: il network riconverge su di una nuova longest chain

È teoricamente possibile che un fork si estenda fino a due blocchi, se due blocchi si trovano quasi contemporaneamente da minatori su "lati" opposti di un fork precedente. Tuttavia, la possibilità che ciò accada è molto bassa. Mentre un fork a un blocco potrebbe verificarsi ogni settimana, una fork a due blocchi è estremamente raro.

L'intervallo di blocco di Bitcoin di 10 minuti è un compromesso di progettazione tra i tempi di conferma rapidi (liquidazione delle transazioni) e la probabilità di un fork. Un tempo di blocco più veloce renderebbe le transazioni più veloci, ma porterà a fork sulla blockchain più frequenti, mentre un tempo di blocco più lento ridurrebbe il numero di fork, ma rallenterebbe le transazioni più lente.

Il Mining e la Gara di Hashing

Il mining di Bitcoin è un settore estremamente competitivo. Il potere di hashing è aumentato esponenzialmente ogni anno dell'esistenza di bitcoin. Da qualche anno la crescita ha riflesso un cambiamento completo della tecnologia, come nel 2010 e nel 2011, quando molti minatori passarono dall'uso del mining della CPU a al mining con le GPU e successivamente al mining field programmable gate array (FPGA). Nel 2013 l'introduzione di il mining con ASIC ha portato a un altro gigantesco balzo in avanti nel settore del mining, collocando la funzione SHA256 direttamente su chip di silicio specializzati ai fini del mining. I primi chip di questo tipo potrebbero fornire più potenza di estrazione in un singolo box rispetto all'intera rete bitcoin nel 2010.

La lista seguente mostra la potenza totale di hashing del network bitcoin, durante i primi cinque anni di operatività:

2009

0.5 MH/sec–8 MH/sec (16x di crescita)

2010

8 MH/sec–116 GH/sec (14,500x di crescita)

2011

16 GH/sec–9 TH/sec (562x di crescita)

2012

9 TH/sec–23 TH/sec (2.5x di crescita)

2013

23 TH/sec–10 PH/sec (450x di crescita)

2014

10 PH/sec–150 PH/sec in August (15x di crescita)

Nel grafico in Potenza di hashing totale, gigahash per secondo, nel corso di due anni, vediamo l'aumento della potenza hashing della rete bitcoin negli ultimi due anni. Come potete vedere, la competizione tra i minatori e la crescita del bitcoin ha comportato un aumento esponenziale della potenza di hashing (hash totale al secondo attraverso la rete).

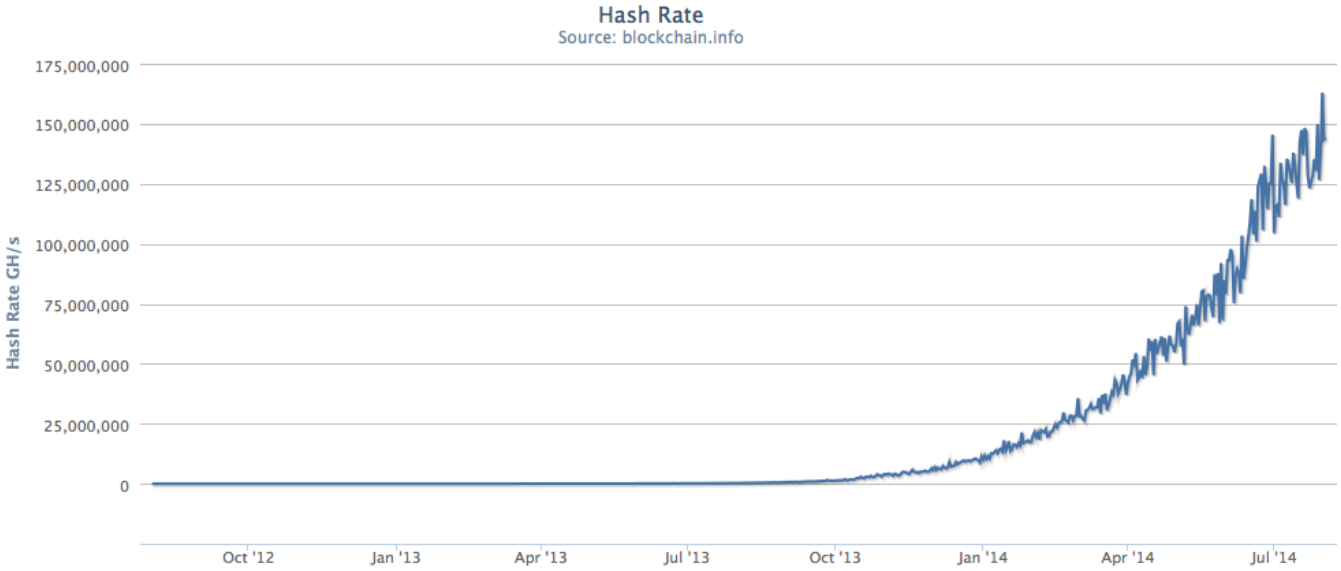

Figure 7. Potenza di hashing totale, gigahash per secondo, nel corso di due anni

Con l'esplosione della quantità di potenza di hashing applicata al mining bitcoin, la difficoltà è aumentata per controbilanciare. La metrica di difficoltà nella tabella mostrata in La metrica di difficoltà di mining di Bitcoin, in due anni è misurata come rapporto tra la difficoltà attuale e la difficoltà minima (la difficoltà del primo blocco) .

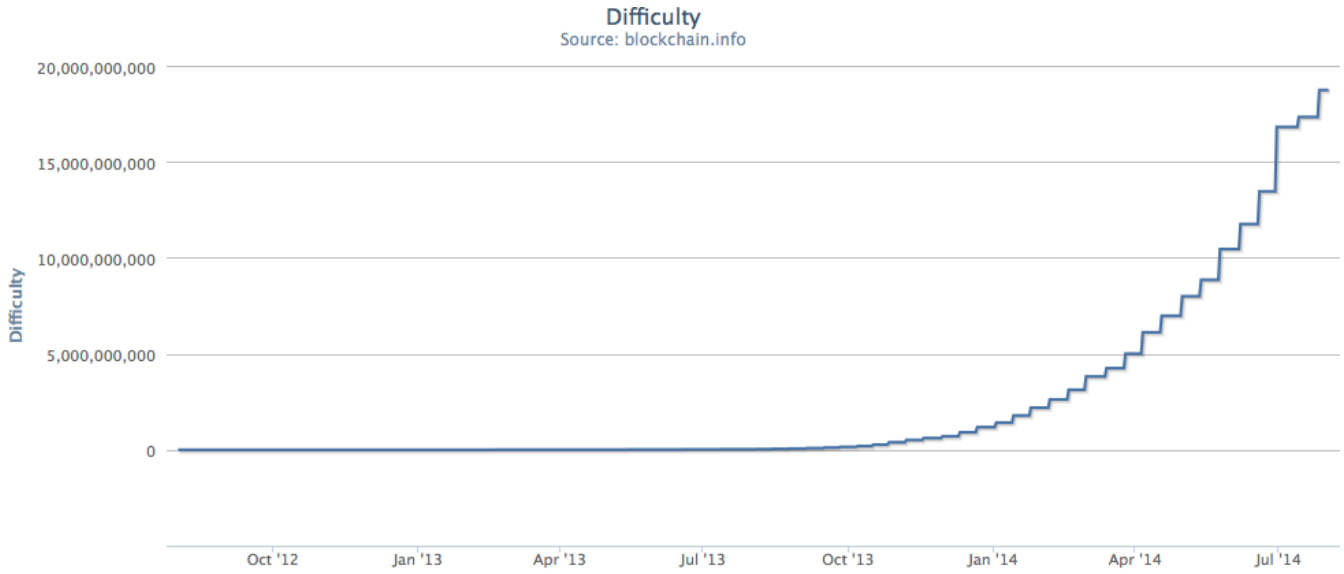

Figure 8. La metrica di difficoltà di mining di Bitcoin, in due anni

Negli ultimi due anni, i chip ASIC per il mining sono diventati sempre più densi, avvicinandosi alla tecnologia di fabbricazione del silicio con una dimensione (risoluzione) di 22 nanometri (nm). Attualmente, i produttori ASIC puntano a sorpassare i produttori di chip CPU generici, progettando chip con dimensioni di 16 nm, perché la redditività del settore del mining sta guidando questo settore ancora più rapidamente del calcolo generale. Non ci sono più balzi giganteschi nell'industria del mining del bitcoin, perché l'industria ha raggiunto la prima linea della Legge di Moore, che stabilisce che la densità di calcolo raddoppierà all'incirca ogni 18 mesi. Tuttavia, la potenza di mining della rete continua ad avanzare ad un ritmo esponenziale mentre la corsa per i chip ad alta densità è abbinata ad una gara per data center ad alta densità dove migliaia di questi chip possono essere schierati. Non si tratta più di quanta estrazione può essere fatta con un solo chip, ma di quanti chip possono essere stipati in un edificio, continuando a dissipare il calore e fornendo una potenza adeguata.

La Soluzione del Nonce Extra

Dal 2012, il bitcoin mining si è evoluto per risolvere una limitazione fondamentale nella struttura dell'intestazione del blocco. Agli albori del bitcoin, un miner poteva trovare un blocco iterando attraverso il nonce fino a quando l'hash risultante era inferiore all'obiettivo. A mano a mano che aumentava la difficoltà, i miner spesso ciclavano tutti i 4 miliardi di valori del nonce senza trovare un blocco. Tuttavia, questo è stato facilmente risolto aggiornando il timestamp del blocco per tenere conto del tempo trascorso. Poiché il timestamp fa parte dell'intestazione, la modifica consentirebbe ai miner di scorrere nuovamente i valori del nonce con risultati diversi. Una volta che la potenza di calcolo dell'hardware di mining ha superato i 4 GH/sec, tuttavia, questo approccio è diventato sempre più difficile perché i valori nonce sono stati esauriti in meno di un secondo. Poiché le attrezzature di mining ASIC iniziarono a spingere e quindi a superare la frequenza hash di 1 TH/sec, il software di mining aveva bisogno di più spazio per i valori nonce al fine di trovare blocchi validi. Il timestamp potrebbe essere allungato un po', ma spostandolo troppo lontano nel futuro potrebbe rendere il blocco invalido. Nell'intestazione del blocco era necessaria una nuova fonte di "modifica". La soluzione era usare la transazione coinbase come fonte di valori extra nonce. Poiché lo script coinbase può archiviare tra 2 e 100 byte di dati, i minatori hanno iniziato a utilizzare quello spazio come spazio extra nonce, consentendo loro di esplorare una gamma molto più ampia di valori di intestazione di blocco per trovare blocchi validi. La transazione coinbase è

inclusa nel merkle tree, il che significa che qualsiasi modifica nello script coinbase fa cambiare la merkle root. Otto byte di extra nonce, più i 4 byte di "standard" nonce permettono ai miner di esplorare un totale di 2^{96} (8 seguito da 28 zeri) possibilità *al secondo* senza dover modificare il timestamp. Se, in futuro, i minatori potrebbero attraversare tutte queste possibilità, potrebbero quindi modificare il timestamp. C'è anche più spazio nello script coinbase per l'espansione futura dello spazio extra nonce.

Le Mining Pool

In questo ambiente altamente competitivo, i singoli miner che lavorano da soli (noti anche come minatori solisti) non hanno alcuna possibilità. La probabilità che trovino un blocco per compensare i costi dell'elettricità e dell'hardware è talmente bassa da rappresentare un gioco d'azzardo, come giocare alla lotteria. Persino il più più veloce sistema di estrazione ASIC, non riesce a stare al passo con i sistemi commerciali che accumulano decine di migliaia di questi chip in magazzini giganteschi vicino a centrali idroelettriche. I miner ora collaborano per formare mining pools, unendo il loro potere di hashing e condividendo la ricompensa tra migliaia di partecipanti. Partecipando ad un pool, i miner ricevono una quota minore della ricompensa complessiva, ma in genere vengono premiati ogni giorno, riducendo l'incertezza.

Diamo un'occhiata ad un esempio specifico. Supponiamo che un miner abbia acquistato hardware di mining con una velocità di hashing combinata di 6.000 gigahashes al secondo (GH/s) o 6 TH/s. Nell'agosto 2014 questa attrezzatura costa circa $ 10.000. L'hardware consuma 3 kilowatt (kW) di elettricità quando è in funzione, 72 kWora al giorno, con un costo di $ 7 o $ 8 al giorno in media. Con l'attuale difficoltà di bitcoin, il miner sarà in grado di estrarre un blocco circa una volta ogni 155 giorni, o ogni 5 mesi. Se il minatore trova un blocco unico in quel lasso di tempo, il pagamento di 25 bitcoin, a circa $ 600 per bitcoin, si tradurrà in un unico pagamento di $ 15.000, che coprirà l'intero costo dell'hardware e l'elettricità consumata nel periodo di tempo, lasciando un profitto netto di circa $ 3000. Tuttavia, la possibilità di trovare un blocco in un periodo di cinque mesi dipende dalla fortuna del miner. Potrebbe trovare due blocchi in cinque mesi e fare un profitto molto grande. Oppure potrebbe non trovare un blocco per 10 mesi e subire una perdita finanziaria. Ancor peggio, la difficoltà dell'algoritmo del proof-of-work di bitcoin rischia di salire significativamente in quel periodo, al ritmo attuale di crescita della potenza di hashing, il che significa che il miner ha, al massimo, sei mesi per chiudere in pareggio, prima che l'hardware diventi effettivamente obsoleto e debba essere sostituito da hardware più potente. Se questo miner partecipa a una mining pool, invece di aspettare una ricompensa di $ 15.000 una volta ogni cinque mesi, sarà in grado di guadagnare circa $ 500 a $ 750 a settimana. I pagamenti regolari da una mining pool lo aiuteranno a ammortizzare il costo dell'hardware e dell'elettricità nel tempo senza correre un enorme rischio. L'hardware sarà ancora obsoleto tra sei e nove mesi e il rischio è ancora elevato, ma le entrate sono almeno regolari e affidabili in quel periodo.

Le mining pool coordinano molte centinaia o migliaia di miner, attraverso protocolli specializzati per le mining pool. I singoli miner configurano le proprie apparecchiature di mining per connettersi a un server della pool, dopo aver creato un account. Il loro hardware di mining rimane connesso al server della pool durante l'estrazione, sincronizzando i loro sforzi con gli altri miner. Così, i miner della medesima pool condividono lo sforzo di estrazione di un blocco e quindi ne condividono la ricompensa.

I blocchi vincenti pagano la ricompensa a un indirizzo bitcoin della pool, piuttosto che a singoli

minatori. Il server della pool eseguirà periodicamente i pagamenti agli indirizzi bitcoin dei singoli miner, una volta che la loro quota dei premi abbia raggiunto una determinata soglia. In genere, il server di pool addebita una percentuale dei premi per fornire il servizio di mining pool.

I minatori che partecipano a un pool dividono il lavoro di ricerca di una soluzione per un blocco candidato, guadagnando "azioni" per il loro contributo di calcolo. Il pool di mining imposta un obiettivo di difficoltà inferiore per guadagnare una condivisione, in genere più di 1.000 volte più semplice della difficoltà della rete bitcoin. Quando qualcuno nel pool riesce a estrarre un blocco, il premio viene versato al pool e poi condiviso con tutti i miner in proporzione al numero di azioni che hanno ottenuto contribuendo allo sforzo.

Le piscine sono aperte a tutti i miner, grandi o piccoli, professionisti o dilettanti. Un pool avrà quindi alcuni partecipanti con una singola piccola macchina, e altri con un garage pieno di hardware di fascia alta. Alcuni eseguiranno attività di mining con poche decine di kilowatt di elettricità, altri gestiranno un data center che consuma un megawatt di energia. Come fa un pool minerario a misurare i singoli contributi, in modo da distribuire equamente i premi, senza la possibilità di barare? La risposta è usare l'algoritmo di proof of work di bitcoin per misurare il contributo di ciascun miner, ma impostarlo a una difficoltà inferiore in modo che anche i minatori della pool vincano una quota abbastanza frequentemente in modo che valga la pena contribuire alla pool. Impostando una difficoltà inferiore per guadagnare quote, la pool misura la quantità di lavoro svolto da ciascun miner. Ogni volta che un minatore della piscina trova un hash dell'header del blocco inferiore alla difficoltà del pool, dimostra di aver eseguito il lavoro di hashing per trovare quel risultato. Ancora più importante, il lavoro per trovare le azioni contribuisce, in un modo statisticamente misurabile, allo sforzo complessivo di trovare un hash più basso rispetto all'obiettivo della rete bitcoin. Migliaia di miner che cercano di trovare gli hash di basso valore alla fine ne troveranno uno abbastanza basso da soddisfare il target richiesto della rete bitcoin.

Torniamo all'analogia di un gioco di dadi. Se i giocatori lanciano i dadi con l'obiettivo di ottenere un numero inferiore a quattro (la difficoltà generale della rete), una pool imposterà un obiettivo più facile, contando quante volte i giocatori della pool sono riusciti a ottenere meno di otto. Quando i giocatori della pool lanciano meno di otto (l'obiettivo di condivisione della pool), guadagnano delle quote, ma non vincono la partita perché non raggiungono l'obiettivo di gioco (meno di quattro). I giocatori raggiungeranno più facilmente l'obiettivo della pool, guadagnandoli in modo molto regolare, anche quando non raggiungono l'obiettivo più difficile, cioè vincere la partita. Ogni tanto, uno dei giocatori della pool lancia i dadi ottenendo un numero inferiore a quattro e la pool ottiene quindi la ricompensa. I guadagni possono essere distribuiti ai giocatori della pool in base alle azioni che hanno guadagnato. Anche se l'obiettivo inferiore a otto non permette di vincere, è un modo giusto per misurare i lanci di dadi dei singoli giocatori, e occasionalmente produce un tiro di meno di quattro (ndt che permette alla pool di vincere la partita).

In modo analogo, una mining pool imposterà una difficoltà di pool che garantirà che, un singolo miner del pool, possa trovare l'hash del blocco abbastanza spesso, con una difficoltà del pool inferiore (ndt a quella della rete bitcoin), guadagnando delle quote. Di tanto in tanto, uno di questi tentativi produrrà un hash del blocco inferiore al target della rete bitcoin, rendendolo un blocco valido con la conseguente vittoria dell'intero pool.

Le Pool Gestite

Molti mining pool sono "gestiti", ovvero c'e' una societa' o individuo che gira un pool server. Il

proprietario del pool server e' chiamato il *pool operator* che addebita una una commissione percentuale sui guadagni ai miner del pool.

Il server della pool esegue un software specializzato e un protocollo di mining pool che coordina le attività dei miner appartenteni alla stessa pool. Il server di pool è inoltre connesso a uno o più full node della rete bitcoin e ha accesso diretto a una copia completa del database della blockchain. Ciò consente al server di pool di convalidare blocchi e transazioni per conto dei miner della pool, sollevandoli dal carico di esecuzione di un nodo completo. Per i miner che partecipano ad una pool, questa è un fattore importante, poiché un nodo completo richiede un computer dedicato con almeno 15-20 GB di memoria permanente (harddisk) e almeno 2 GB di memoria (RAM). Inoltre, il software bitcoin in esecuzione sul nodo completo deve essere monitorato, gestito e aggiornato frequentemente. Qualsiasi fermo macchina causato da una mancanza di manutenzione o mancanza di risorse danneggerà la redditività del miner. Per molti miner, la possibilità di estrarre senza un nodo completo è un altro grande vantaggio che si ottiene unendosi ad una pool.

I miner della pool si collegano al server della della pool alla quale appartengono utilizzando un protocollo di mining come Stratum (STM) o GetBlockTemplate (GBT). Un vecchio standard chiamato GetWork (GWK) è diventato obsoleto dalla fine del 2012, perché non supporta facilmente l'estrazione a tassi di hash superiori a 4 GH/s. Entrambi i protocolli STM e GBT creano blocchi *templates* che contengono un modello di un'intestazione di blocco candidato. Il server di pool costruisce un blocco candidato aggregando le transazioni, aggiungendo una transazione coinbase (con spazio extra nonce), calcolando la merkle root e collegando l'hash del blocco precedente. L'intestazione del blocco candidato viene quindi inviata a ciascuno dei miner del pool come modello. Ogni pool minatore quindi utilizza il modello di blocco, a una difficoltà inferiore rispetto alla difficoltà della rete bitcoin, e invia eventuali risultati di successo al server della pool per guadagnare azioni.

P2Pool

Queste pool creano la possibilità di cheating da parte dell'operatore del pool, che potrebbe indirizzare lo sforzo del pool verso le transazioni a doppia spesa o i blocchi invalidati (vedere Attacchi al Consenso). Inoltre, i server di pool centralizzati rappresentano un singolo punto di errore. Se il server della piscina è inattivo o viene rallentato da un attacco denial-of-service, i minatori della piscina non possono estrarlo. Nel 2011, per risolvere questi problemi di centralizzazione, è stato proposto e implementato un nuovo metodo di mining pool: P2Pool è un pool di mining peer-to-peer, senza un operatore centrale.

P2Pool funziona decentralizzando le funzioni del server di pool, implementando un sistema parallelo simile a blockchain chiamato a *catena condivisa*. Una catena di azioni è una blockchain in esecuzione ad una difficoltà inferiore rispetto alla blockchain bitcoin. La catena condivisa consente ai pool di miner di collaborare in un pool decentralizzato, estraendo le azioni della catena ad un tasso di un blocco ogni 30 secondi. Ciascuno dei blocchi della catena di azioni registra un premio proporzionale per i minatori del pool che contribuiscono al lavoro, portando avanti le azioni dal precedente blocco. Quando uno dei blocchi condivisi raggiunge anche l'obiettivo di difficoltà della rete bitcoin, viene propagato e incluso nella blockchain bitcoin, ricompensando tutti i minatori del pool che hanno contribuito a tutte le azioni che hanno preceduto il blocco azionario vincente. Essenzialmente, invece di un server di pool che tiene traccia delle azioni e dei premi del miner, la catena di condivisione consente a tutti i minatori di pool di tenere traccia di tutte le azioni utilizzando un meccanismo di consenso decentralizzato come il meccanismo di consenso

blockchain del bitcoin.

Il mining di P2Pool è più complesso del mining pool (ndt tradizionale) perché richiede che i miner abbiano un computer dedicato con sufficiente spazio su disco, memoria e larghezza di banda Internet per supportare un intero nodo bitcoin e il software del nodo P2Pool. I miniport P2Pool collegano il loro hardware di mining al loro nodo P2Pool locale, che simula le funzioni di un server di pool inviando modelli di blocchi all'hardware di mining. Su P2Pool, i singoli miner del pool costruiscono i propri blocchi candidati, aggregando le transazioni in modo molto simile ai singoli miner, ma in seguito collaborano alla catena di condivisione. P2Pool è un approccio ibrido che ha il vantaggio di pagamenti molto più granulari rispetto al mining solista, ma senza dare troppo controllo a un operatore di pool come i pool gestiti.

Recentemente, la partecipazione a P2Pool è aumentata in modo significativo creando una concentrazione di miner nelle pool tali da creare preoccupazioni per un attacco del 51% (vedi Attacchi al Consenso). L'ulteriore sviluppo del protocollo P2Pool continua con l'aspettativa di rimuovere la necessità di eseguire un nodo completo e quindi rendere ancora più facile l'utilizzo dell'estrazione decentralizzata.

Anche se P2Pool riduce la concentrazione della potenza da parte degli operatori del pool di mining, è presumibilmente vulnerabile agli attacchi del 51% contro la stessa catena di azioni. Un'adozione molto più ampia di P2Pool non risolve il problema di attacco del 51% per bitcoin stesso. Piuttosto, P2Pool rende il bitcoin più solido nel complesso, come parte di un ecosistema minerario diversificato.

Attacchi al Consenso

Il meccanismo di consenso di Bitcoin è, almeno teoricamente, vulnerabile agli attacchi di miner (o pool) che tentano di usare il loro potere di hashing in modo disonesto o distruttivo. Come abbiamo visto, il meccanismo del consenso dipende dal fatto che la maggioranza dei miner agisca onestamente per interesse personale. Tuttavia, se un miner o un gruppo di miner, può raggiungere una quota significativa del potere di calcolo, essi possono attaccare il meccanismo del consenso in modo da interrompere la sicurezza e la disponibilità della rete bitcoin.

È importante notare che gli attacchi di consenso possono influenzare solo il consenso futuro o, nel migliore dei casi, il passato più recente (decine di blocchi). Il registro di Bitcoin diventa sempre più immutabile con il passare del tempo. Mentre in teoria, un fork può essere raggiunto a qualsiasi profondità, nella pratica, la potenza di calcolo necessaria per forzare un fork molto profondo è immensa, rendendo i vecchi blocchi praticamente immutabili. Anche gli attacchi di consenso non influiscono sulla sicurezza delle chiavi private e dell'algoritmo di firma (ECDSA). Un attacco di consenso non può sottrarre bitcoin, spendere bitcoin senza firme, reindirizzare bitcoin o modificare in altro modo transazioni passate o informazioni di proprietà. Gli attacchi di consenso possono interessare solo i blocchi più recenti e causare interruzioni di tipo denial-of-service sulla creazione di blocchi futuri.

Uno scenario di attacco contro il meccanismo del consenso è chiamato "attacco del 51%". In questo scenario un gruppo di minatori, controllando una maggioranza (51%) del potere di hashing della rete totale, collude per attaccare bitcoin. Con la possibilità di estrarre la maggior parte dei blocchi, i minatori che tentano un attacco possono causare "fork" deliberati nelle transazioni blockchain e double-spending o eseguire attacchi denial-of-service contro specifiche transazioni o indirizzi. Un

fork/double spending attack è quello in cui l'attaccante fa sì che i blocchi precedentemente confermati vengano invalidati mediante la biforcazione sotto di essi e la ricomposizione in una catena alternativa. Con una potenza di calcolo sufficiente, un utente malintenzionato può invalidare sei o più blocchi di fila, causando l'annullamento delle transazioni considerate immutabili (sei conferme). Tieni presente che una doppia spesa può essere eseguita solo sulle transazioni dell'hacker, per le quali l'attaccante può produrre una firma valida. Doppio-spendere le proprie transazioni è redditizio se invalidando una transazione l'attaccante può ottenere un pagamento non reversibile o un prodotto senza pagarlo.

Esaminiamo un esempio pratico di un attacco del 51%. Nel primo capitolo, abbiamo esaminato una transazione tra Alice e Bob per una tazza di caffè. Bob, il proprietario del caffè, è disposto ad accettare pagamenti per tazze di caffè senza attendere conferma (estrazione in un blocco), perché il rischio di una doppia spesa per una tazza di caffè è basso rispetto alla convenienza del servizio clienti rapido . Questo è simile alla pratica dei coffee shop che accettano pagamenti con carta di credito senza firma per importi inferiori a $ 25, perché il rischio di un chargeback di una carta di credito è basso mentre il costo di ritardare la transazione per ottenere una firma è relativamente più grande. Al contrario, la vendita di un articolo più costoso per bitcoin comporta il rischio di un attacco a doppia spesa, in cui l'acquirente trasmette una transazione concorrente che spende gli stessi input (UTXO) e annulla il pagamento al commerciante. Un attacco a doppia spesa può avvenire in due modi: o prima che una transazione sia confermata, o se l'attaccante sfrutta un fork blockchain per annullare diversi blocchi. Un attacco del 51% consente agli aggressori di raddoppiare le proprie transazioni nella nuova catena, annullando la transazione corrispondente nella vecchia catena.

Nel nostro esempio, l'aggressore malvagio Mallory si reca alla galleria di Carol e acquista un bellissimo dipinto trittico raffigurante Satoshi Nakamoto come Prometeo. Carol vende dipinti "The Great Fire" per $ 250.000 in bitcoin, a Mallory. Invece di aspettare sei o più conferme sulla transazione, Carol avvolge e consegna i dipinti a Mallory dopo una sola conferma. Mallory lavora con un complice, Paul, che gestisce una grande mining pool e il complice lancia un attacco del 51% non appena la transazione di Mallory è inclusa in un blocco. Paul ordina al pool di miner di minare nuovamente un blocco con la stessa altezza del blocco contenente la transazione di Mallory, sostituendo il pagamento di Mallory a Carol con una transazione che duplica lo stesso input del pagamento di Mallory. La transazione a doppia spesa consuma lo stesso UTXO e la ripaga sul portafoglio di Mallory, invece di pagarla a Carol, essenzialmente permettendo a Mallory di mantenere il bitcoin. Paul quindi indirizza la pool a estrarre un ulteriore blocco, in modo da rendere la catena contenente la transazione a doppia spesa più lunga della catena originale (causando un fork sotto il blocco contenente la transazione di Mallory). Quando la biforcazione della blockchain si risolve a favore della nuova (più lunga) catena, la transazione a doppio spesa sostituisce il pagamento originale a Carol. A Carol mancano ora i tre dipinti e inoltre non ha ottenuto il pagamenti con bitcoin. Durante tutta questa attività, i partecipanti alla pool di Paul potrebbero rimanere beatamente inconsapevoli del tentativo di doppia spesa, perché sono miner automatizzati e non possono monitorare ogni transazione o blocco.

Per proteggersi da questo tipo di attacco, un commerciante che vende oggetti di grande valore deve attendere almeno sei conferme prima di dare il prodotto all'acquirente. In alternativa, il commerciante deve utilizzare un conto di deposito a garanzia multi-firma, in attesa di diverse conferme dopo che l'account di deposito a garanzia è stato finanziato. Più conferme si accumulano, più diventa difficile invalidare una transazione con un attacco del 51%. Per gli articoli di valore

elevato, il pagamento tramite bitcoin sarà comunque conveniente ed efficiente anche se l'acquirente deve attendere 24 ore per la consegna, il che corrisponderebbe a circa 144 conferme.

Oltre a un attacco a doppia spesa, l'altro scenario per un attacco di consenso consiste nel negare il servizio a partecipanti specifici di bitcoin (specifici indirizzi bitcoin). Un attaccante con la maggioranza del potere di mining può semplicemente ignorare transazioni specifiche. Se sono inclusi in un blocco estratto da un altro miner, l'utente malintenzionato può deliberatamente eseguire il fork e minare nuovamente quel blocco, escludendo di nuovo le transazioni specifiche. Questo tipo di attacco può comportare un rifiuto prolungato del servizio contro un indirizzo specifico o un insieme di indirizzi per tutto il tempo in cui l'attaccante controlla la maggior parte della potenza di hashing.

Nonostante il suo nome, lo scenario di attacco del 51% non richiede in realtà il 51% della potenza di hashing. In realtà, un simile attacco può essere tentato con una percentuale minore della potenza di hashing. La soglia del 51% è semplicemente il livello al quale un tale attacco è quasi garantito per avere successo. Un attacco di consenso è essenzialmente un tiro alla fune per il prossimo blocco e il gruppo "più forte" ha più probabilità di vincere. Con meno potere di hashing, la probabilità di successo è ridotta, perché altri minatori controllano la generazione di alcuni blocchi con la loro "onesta" potenza di calcolo. Un modo per vederlo è che più potere di hashing ha un attaccante, più lungo è il fork che può creare deliberatamente, più blocchi nel recente passato possono essere invalidati, o più blocchi in futuro potranno essere controllati. I gruppi di ricerca sulla sicurezza hanno utilizzato modelli statistici per affermare che vari tipi di attacchi di consenso sono possibili con un minimo del 30% della potenza di hashing.

Il massiccio aumento della potenza di hashing totale ha probabilmente reso i bitcoin impermeabili agli attacchi di un singolo miner. Non esiste un modo per un miner solitario di controllare più di una piccola percentuale della potenza di estrazione totale. Tuttavia, la centralizzazione del controllo causata dalle mining pool ha introdotto il rischio di attacchi di profitto da parte di un gestore di pool. L'operatore del pool, in un pool gestito, controlla la costruzione dei blocchi candidati e controlla anche quali transazioni sono incluse. Ciò conferisce al gestore del pool, il potere di escludere le transazioni o introdurre transazioni a doppia spesa. Se tale abuso di potere è fatto in modo limitato e sottile, un operatore di pool potrebbe teoricamente trarre profitto da un attacco di consenso senza essere notato.

Tuttavia, non tutti gli aggressori saranno motivati dal profitto. Uno scenario di potenziale attacco è dove un attaccante intende disturbare la rete bitcoin senza la possibilità di trarre profitto da tale disturbo. Un attacco malevolo mirato a paralizzare bitcoin richiederebbe enormi investimenti e pianificazione segreta, ma potrebbe essere probabilmente lanciato da un attaccante ben finanziato e molto probabilmente sponsorizzato dallo stato. In alternativa, un aggressore ben finanziato potrebbe attaccare il consenso di bitcoin accumulando simultaneamente hardware di data mining, compromettendo gli operatori di pool e attaccando altri pool con denial-of-service. Tutti questi scenari sono teoricamente possibili, ma sempre più impraticabili in quanto la potenza di hashing complessiva della rete bitcoin continua a crescere esponenzialmente.

Indubbiamente, un serio attacco di consenso eroderebbe la fiducia nel bitcoin a breve termine, causando probabilmente un significativo calo dei prezzi. Tuttavia, la rete bitcoin e il software sono in continua evoluzione, quindi gli attacchi di consenso verrebbero affrontati con contromisure immediate da parte della comunità bitcoin, rendendo il bitcoin più solido, subdolo e robusto che mai.